tutorialr.com

tutorial

Windows App SDK

Theme Transition

tutorialr.com

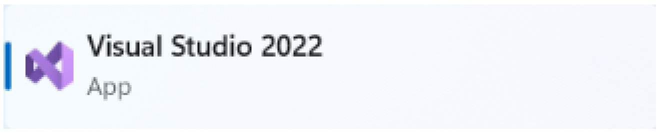
Theme Transition

Theme Transition shows how you can create **Transitions** that apply to elements within an application using the **Windows App SDK**.

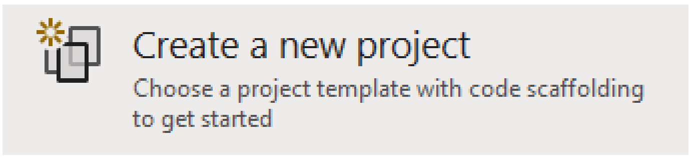
Step 1

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

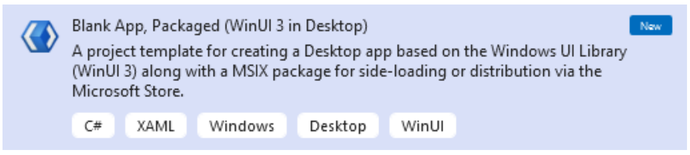
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



Once **Visual Studio 2022** has started select **Create a new project**.

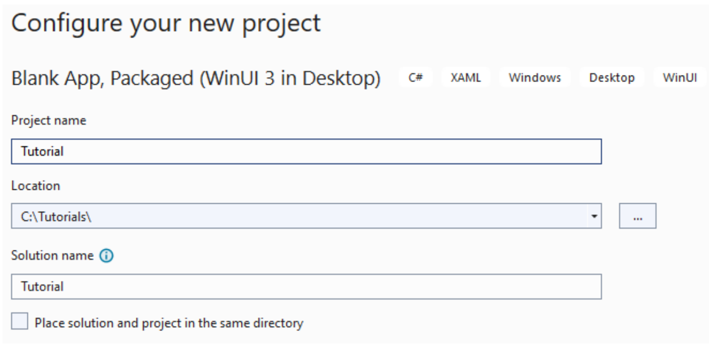

Create a new project
Choose a project template with code scaffolding to get started

Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.


Blank App, Packaged (WinUI 3 in Desktop)

A project template for creating a Desktop app based on the Windows UI Library (WinUI 3) along with a MSIX package for side-loading or distribution via the Microsoft Store.

After that in **Configure your new project** type in the **Project name** as *ThemeTransition*, then select a Location and then select **Create** to start a new **Solution**.


Configure your new project

Blank App, Packaged (WinUI 3 in Desktop)

Project name: Tutorial

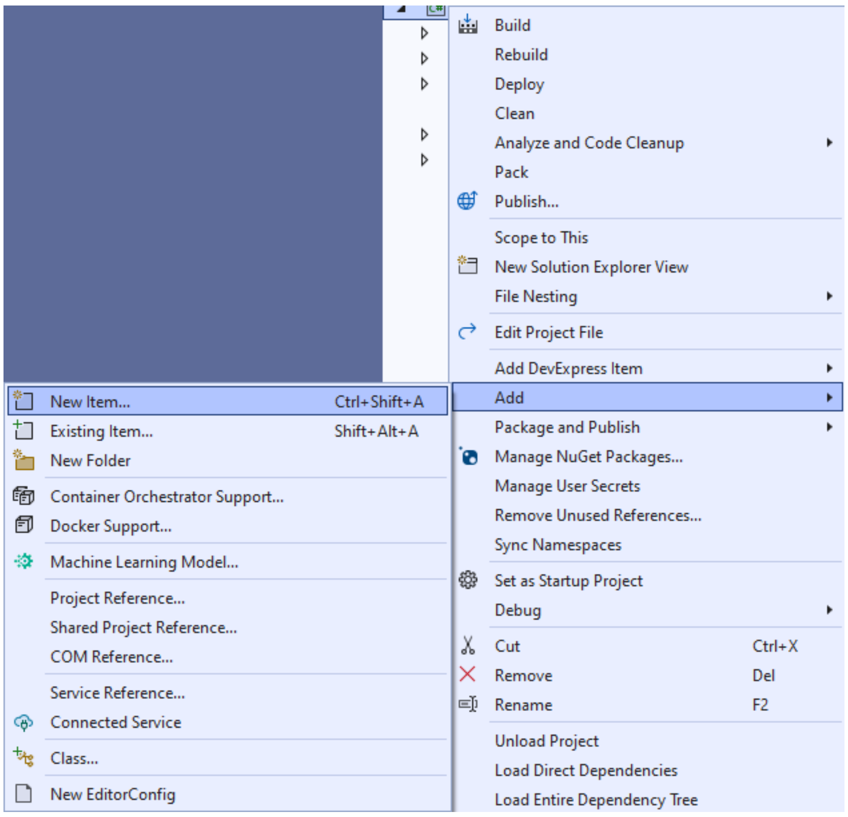
Location: C:\Tutorials\

Solution name: Tutorial

Place solution and project in the same directory

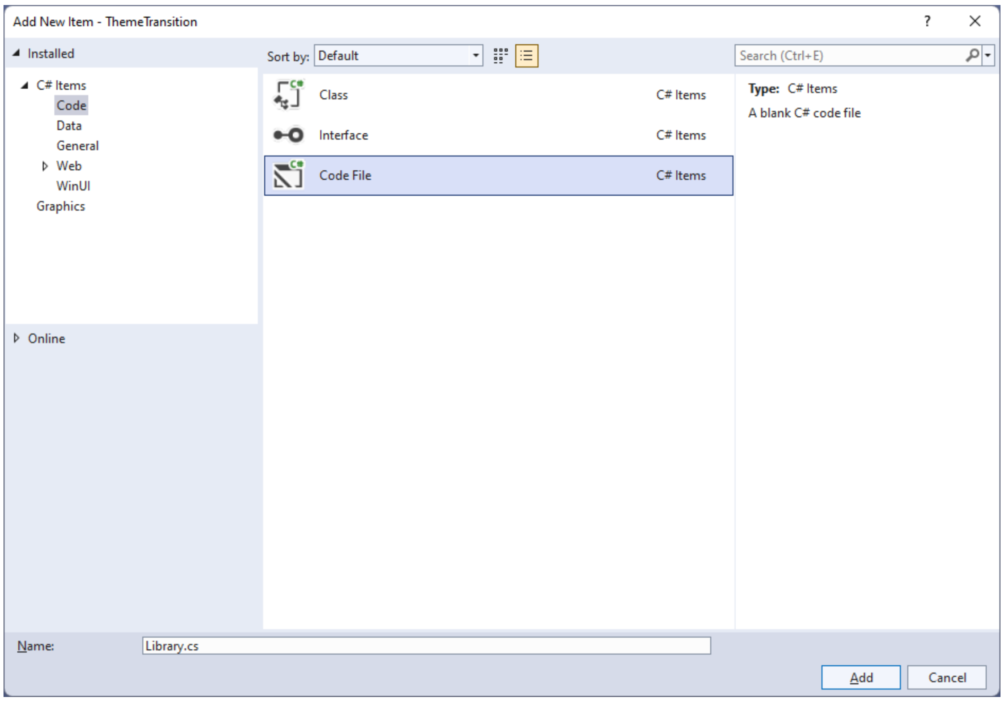
Step 2

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…**



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 *Library.cs* and then **Click** on **Add**.



Type: C# Items
A blank C# code file

Name: Library.cs

Step 4

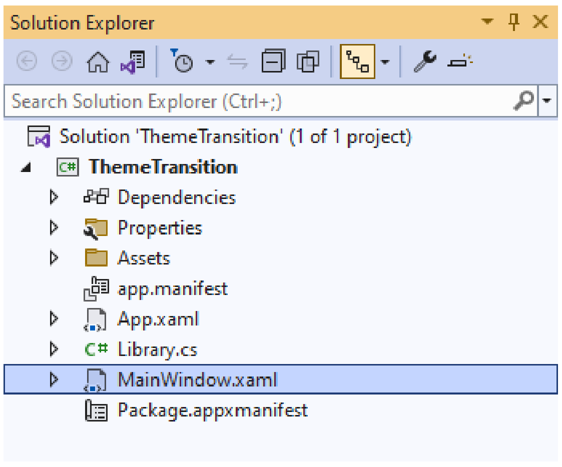
You will now be in the **View** for the **Code** of *Library.cs*, within this type the following **Code**:

```
using Microsoft.UI;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Shapes;
using System.Collections.Generic;
using System.Linq;
using Windows.UI;

internal class Library
{
    private static readonly List<Color> _colours = new()
    {
        Colors.Black, Colors.Gray, Colors.Red, Colors.Orange, Colors.Yellow,
        Colors.Green, Colors.Cyan, Colors.Blue, Colors.Magenta, Colors.Purple
    };

    private static int _index = 0;

    public static void Add(StackPanel panel)
    {
        Rectangle previous = panel.Children.LastOrDefault() as Rectangle;
        if(previous == null || _index == _colours.Count)
        {
            _index = 0;
        }
        panel.Children.Add(new Rectangle()
        {
            Width = 50,
            Height = 50,
            Fill = new SolidColorBrush(_colours[_index])
        });
        _index++;
    }

    public static void Remove(StackPanel panel)
    {
        int count = panel.Children.Count;
        if (count > 0)
        {
            panel.Children.RemoveAt(count - 1);
        }
    }
}
```

The **`Class`** that has been defined in *Library.cs* has **`List`** of **`Color`** for the colours to use for elements along with an **`int`** value for keeping track of which one will be used. Then there is a **Method** of **`Add`**, which will add a **`Rectangle`** to a **`StackPanel`** with the given next colour to use and the **Method** of **`Remove`**, which will remove added **`Rectangle`** elements from a **`StackPanel`**.

Step 5

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



Step 6

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 7

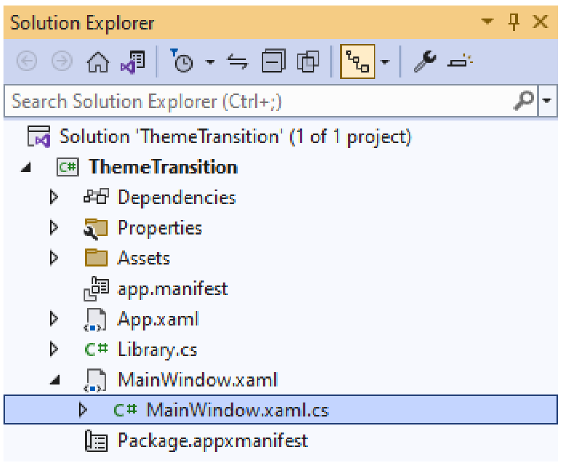
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<Grid>
    <Viewbox Margin="25">
        <StackPanel Name="Display" Spacing="5"
            Orientation="Horizontal" HorizontalAlignment="Center">
            <StackPanel.ChildrenTransitions>
                <TransitionCollection>
                    <EntranceThemeTransition IsStaggeringEnabled="True" />
                </TransitionCollection>
            </StackPanel.ChildrenTransitions>
        </StackPanel>
    </Viewbox>
    <CommandBar VerticalAlignment="Bottom" HorizontalAlignment="Stretch" >
        <AppBarButton Icon="Add" Label="Add" Click="Add_Click"/>
        <AppBarButton Icon="Remove" Label="Remove" Click="Remove_Click"/>
    </CommandBar>
</Grid>
```

This **XAML** features a **`Grid`** with a **`ViewBox`** to **Scale** elements and in this is a **`StackPanel`** which has the **Theme Transition** set for the **`ChildrenTransitions`** of **`EntranceThemeTransition`** with an **`AppBarButton`** to *Add* and *Remove* elements.

Step 8

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**.


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'ThemeTransition' (1 of 1 project)
  ThemeTransition
    Dependencies
    Properties
    Assets
    app.manifest
    App.xaml
    Library.cs
    MainWindow.xaml
      MainWindow.xaml.cs
    Package.appxmanifest
```

Step 9

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 10

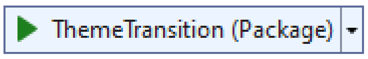
Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

```
private void Add_Click(object sender, RoutedEventArgs e)
{
    Library.Add(Display);
}

private void Remove_Click(object sender, RoutedEventArgs e)
{
    Library.Remove(Display);
}
```

The **Methods** of **`Add_Click`** and **`Remove_Click`** will when **Clicked** will call the **Methods** within *Library.cs* of **`Add`** and **`Remove`** from **`Library`** passing in the **`StackPanel`**.

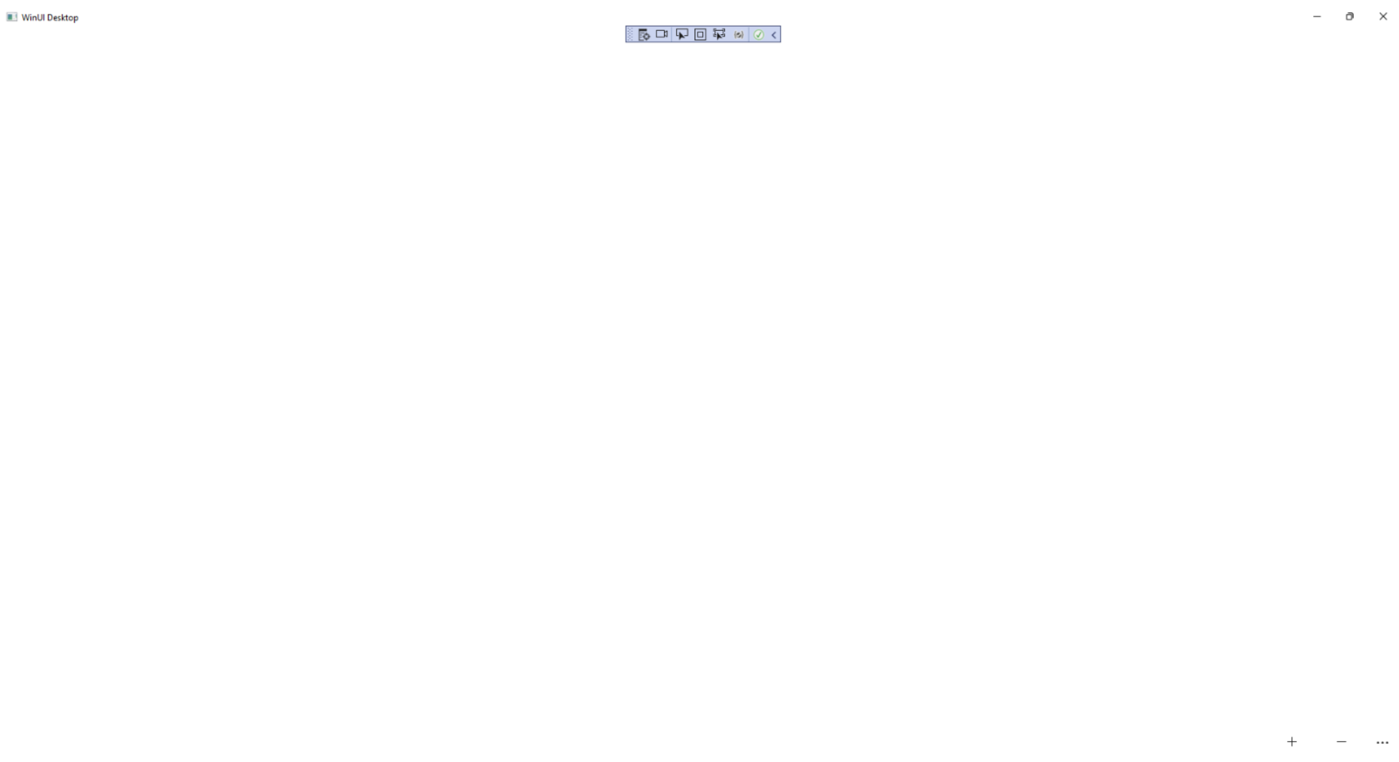
Step 11

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



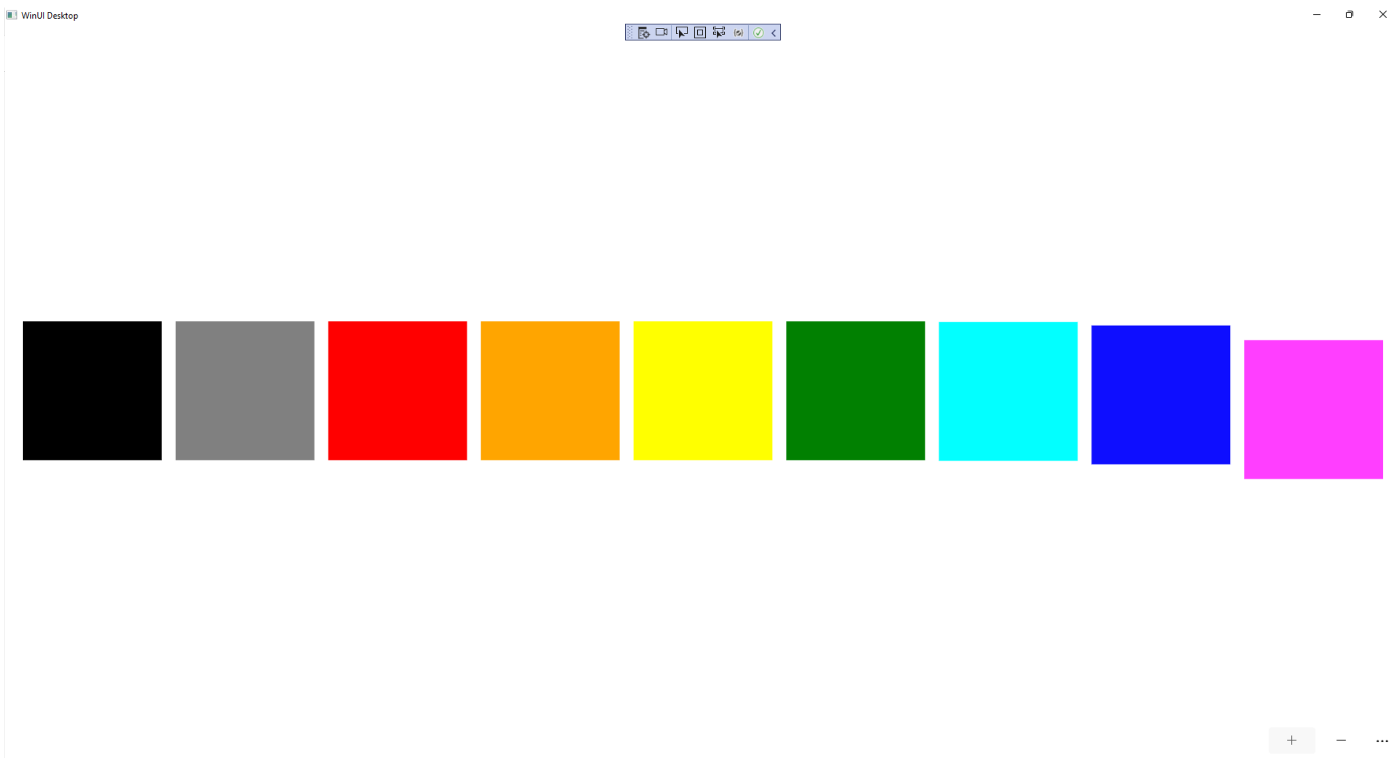
Step 12

Once running you should see the **`AppBarButton`** for *Add* and *Remove*.



Step 13

You can select the **`AppBarButton`** for *Add* to add some elements and they should appear with a **Theme Transition**.



Step 14

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!