



Windows App SDK



System Backdrops

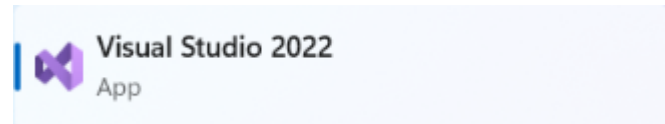
System Backdrops

System Backdrops shows how you can use **SystemBackdrops** within the **Window** of an application using the **Windows App SDK**.

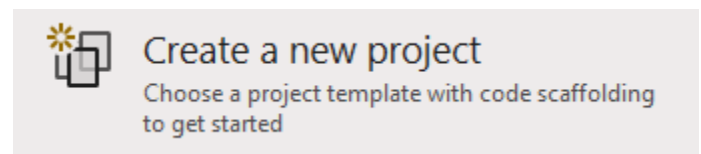
Step 1

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

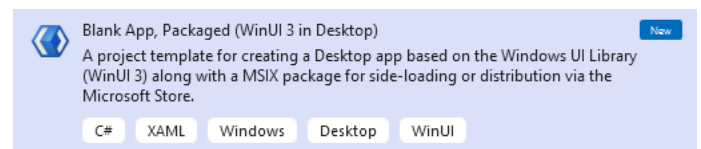
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



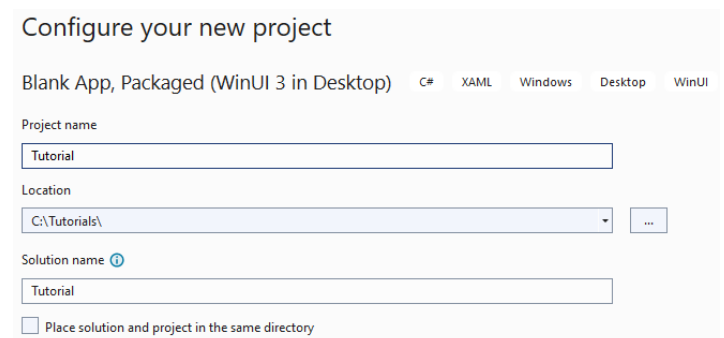
Once **Visual Studio 2022** has started select **Create a new project**.



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

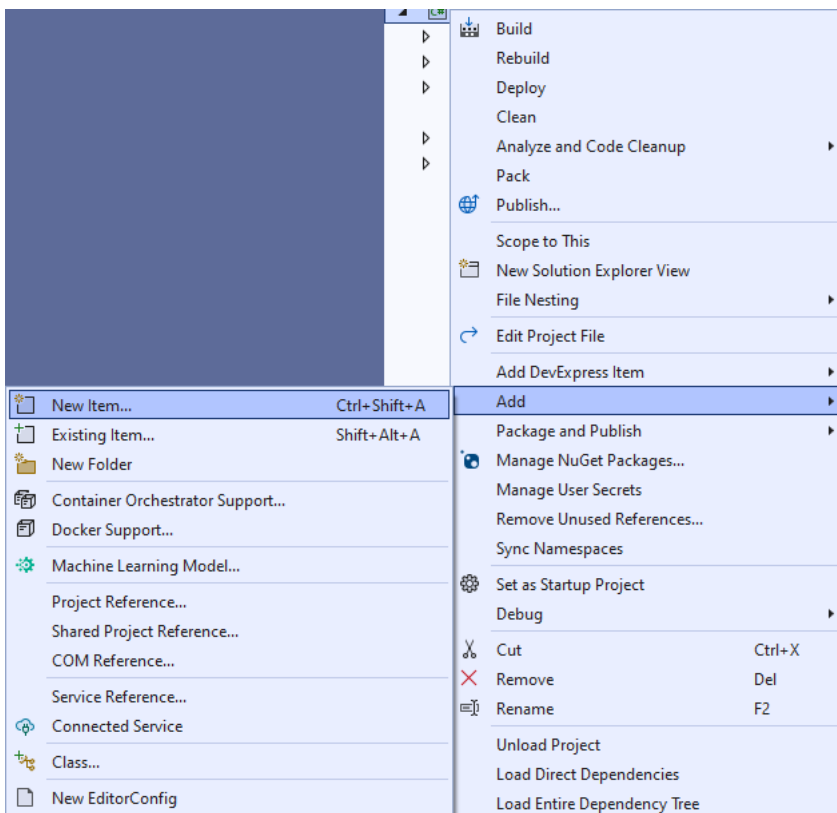


After that in **Configure your new project** type in the **Project name** as *SystemBackdrops*, then select a Location and then select **Create** to start a new **Solution**.



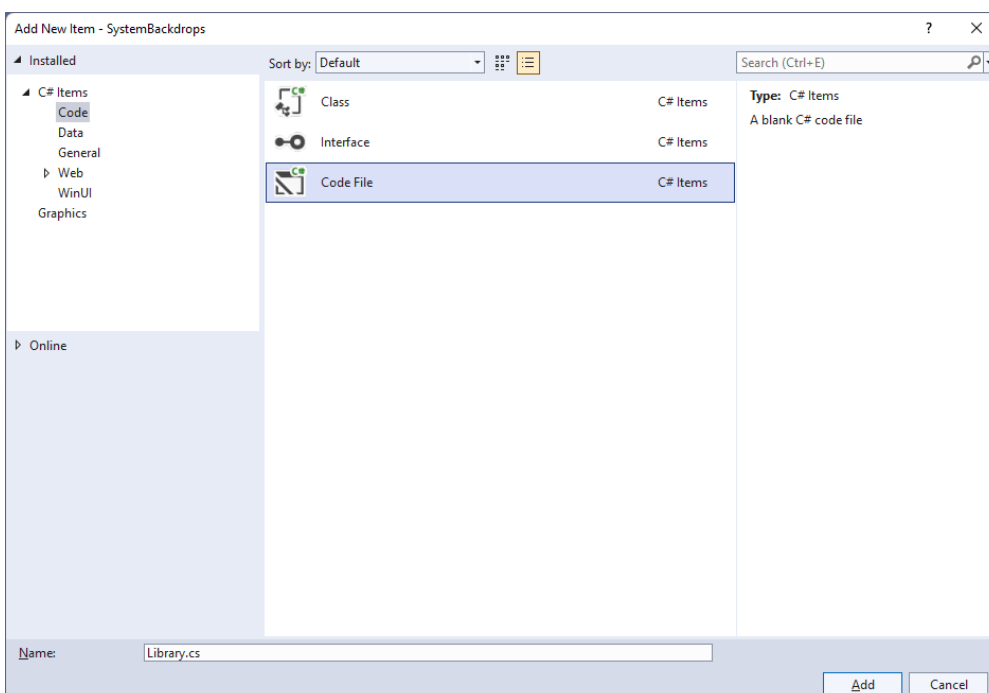
Step 2

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**.



Step 4

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

```
using Microsoft.UI.Composition;
using Microsoft.UI.Composition.SystemBackdrops;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using System.Runtime.InteropServices;
using Windows.System;
using WinRT;

internal class Library
{
    private object _queue;
    private ISystemBackdropControllerWithTargets _controller;

    [StructLayout(LayoutKind.Sequential)]
    struct DispatcherQueueOptions
    {
        internal int dwSize;
        internal int threadType;
        internal int apartmentType;
    }

    [DllImport("CoreMessaging.dll")]
    private static extern int CreateDispatcherQueueController(
        [In] DispatcherQueueOptions options,
        [In, Out, MarshalAs(UnmanagedType.IUnknown)]
        ref object dispatcherQueueController);

    private void EnsureDispatcherQueueController()
    {
        if (DispatcherQueue.GetForCurrentThread() != null)
            return;
        if (_queue == null)
        {
            DispatcherQueueOptions options;
            options.dwSize = Marshal.SizeOf(typeof(DispatcherQueueOptions));
            options.threadType = 2; // DQTYPE_THREAD_CURRENT
            options.apartmentType = 2; // DQTAT_COM_STA
            _ = CreateDispatcherQueueController(options, ref _queue);
        }
    }

    // Set Backdrop
}
```

The **Class** that has been defined so far in *Library.cs* has a **Member** for an **object** and an **Interface** of **ISystemBackdropControllerWithTargets** then there is some code to make the **System Backdrops** work correctly. There is a **struct** that will be used with the **Method** which will use a **DllImport** for some **Unmanaged Code** that is part of the **API** for **Windows** to use **CreateDispatcherQueueController**. This is called from the **Method** which will configure a **DispatcherQueue** which will ensure the **System Backdrops** work as needed of **EnsureDispatcherQueueController**.

Step 5

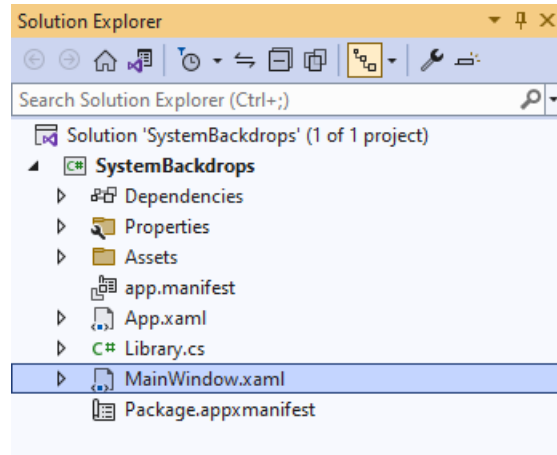
While still in the **Class** for *Library.cs* and after the **Comment** of `// Set Backdrop` type in the following **Method**:

```
public void SetBackdrop(Window window, ComboBox options)
{
    if (_controller != null)
        _controller.Dispose();
    EnsureDispatcherQueueController();
    string value = (options.SelectedItem as ComboBoxItem).Content as string;
    switch (value)
    {
        case "Acrylic":
            if (DesktopAcrylicController.IsSupported())
            {
                _controller = new DesktopAcrylicController();
                _controller.AddSystemBackdropTarget(
                    window.As<ICompositionSupportsSystemBackdrop>());
                _controller.SetSystemBackdropConfiguration(
                    new SystemBackdropConfiguration());
            }
            break;
        case "Mica":
            if (MicaController.IsSupported())
            {
                _controller = new MicaController();
                _controller.AddSystemBackdropTarget(
                    window.As<ICompositionSupportsSystemBackdrop>());
                _controller.SetSystemBackdropConfiguration(
                    new SystemBackdropConfiguration());
            }
            break;
        default:
            _controller = null;
            break;
    }
}
```

This **Method** will check if **ISystemBackdropControllerWithTargets** has been set, or is not **null** then it will call the **Method** for **EnsureDispatcherQueueController** then it will get the value from the **ComboBox** that was passed in, then within a **switch** Statement, there is an option for *Acrylic* and *Mica* which is one of the **System Backdrops** that can be supported, to check this is supported the **Method** of **IsSupported()** of the **Class** of **DesktopAcrylicController** or **MicaController** is called, then the **Method** of **AddSystemBackdropTarget** and **SetSystemBackdropConfiguration** is used to apply the **System Backdrop** with **ISystemBackdropControllerWithTargets** to the **Window** that was passed in and the **default** option of the **switch** will set the **ISystemBackdropControllerWithTargets** to **null**.

Step 6

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



Step 7

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **StackPanel1**, 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 8

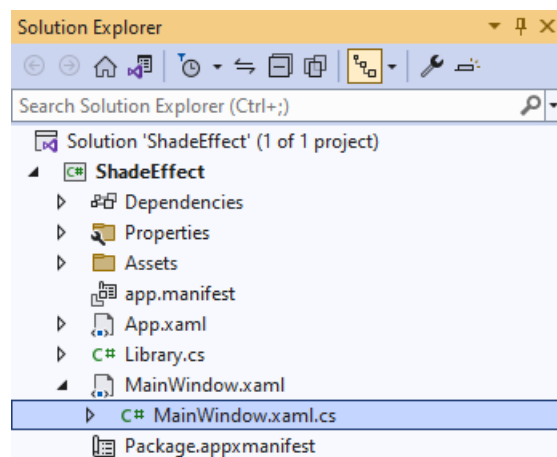
While still in the **XAML** for **MainWindow.xaml** above `</Window>`, type in the following **XAML**:

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*/>
  </Grid.RowDefinitions>
  <ComboBox Grid.Row="0" Name="Options" Margin="25" HorizontalAlignment="Stretch"
    SelectionChanged="Options_SelectionChanged">
    <ComboBoxItem IsSelected="True">None</ComboBoxItem>
    <ComboBoxItem>Acrylic</ComboBoxItem>
    <ComboBoxItem>Mica</ComboBoxItem>
  </ComboBox>
  <Viewbox Grid.Row="1">
    <StackPanel Spacing="5" Orientation="Horizontal"
      HorizontalAlignment="Center">
      <Rectangle Width="50" Height="50" Fill="Black"/>
      <Rectangle Width="50" Height="50" Fill="Gray"/>
      <Rectangle Width="50" Height="50" Fill="Red"/>
      <Rectangle Width="50" Height="50" Fill="Orange"/>
      <Rectangle Width="50" Height="50" Fill="Yellow"/>
      <Rectangle Width="50" Height="50" Fill="Green"/>
      <Rectangle Width="50" Height="50" Fill="Cyan"/>
      <Rectangle Width="50" Height="50" Fill="Blue"/>
      <Rectangle Width="50" Height="50" Fill="Magenta"/>
      <Rectangle Width="50" Height="50" Fill="Purple"/>
    </StackPanel>
  </Viewbox>
</Grid>
```

This **XAML** features a **Grid** with two **Rows**, the first **Row** is for a **ComboBox** which contains the name of the **System Backdrops** and *None* that will be used to apply the selected **System Backdrop**. The second **Row** is a **ViewBox** which is used to **Scale** elements and in this is a **StackPanel** that contains **Rectangle** elements.

Step 9

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 10

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 11

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

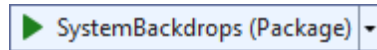
```
private readonly Library _library = new();

private void Options_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    _library.SetBackdrop(this, Options);
}
```

The **Method** of **Options_SelectionChanged** will call the **Method** within *Library.cs* of **SetBackdrop** from an **Instance** of **Library** called **_library** created with **new()** and will also pass in the current **Window** with the **Keyword** of **this** which will pass in the current **Instance** of the **Window**.

Step 12

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



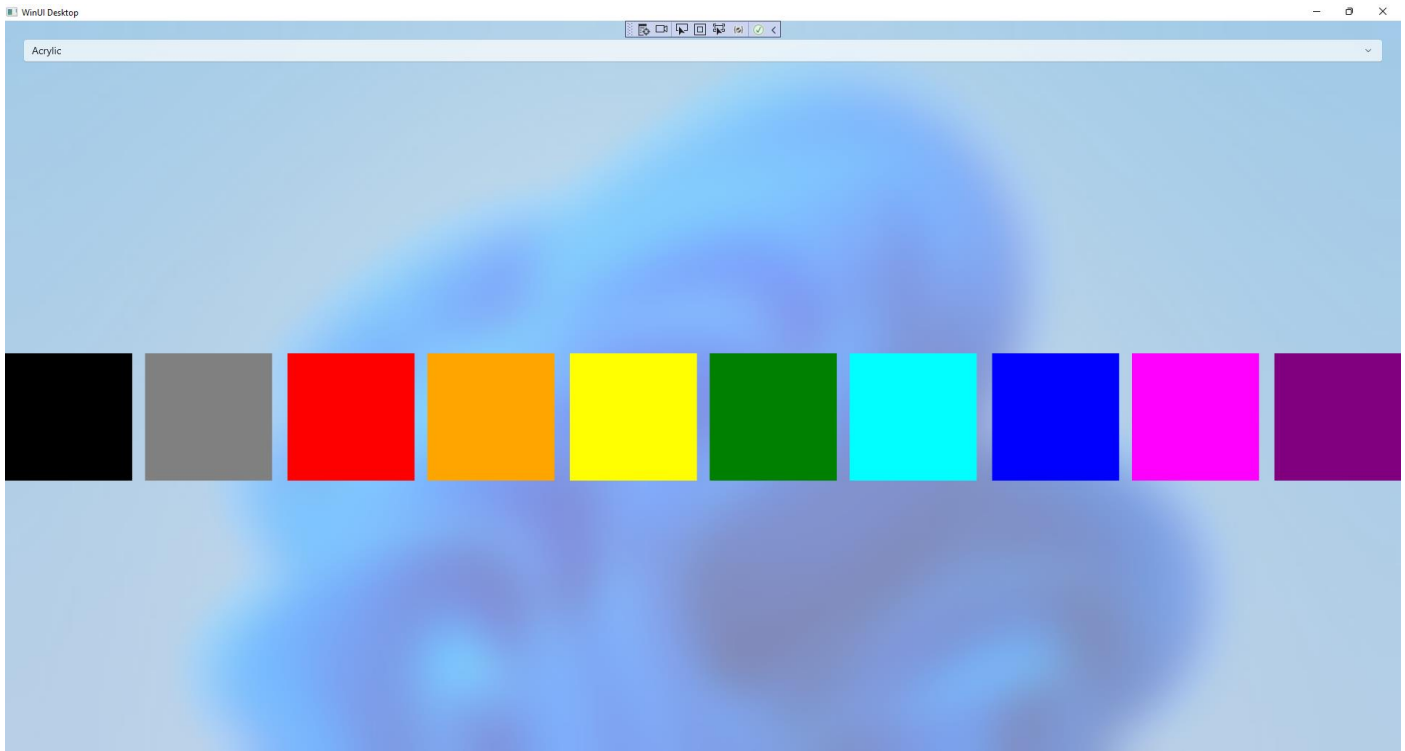
Step 13

Once running you should a **Combobox** with **System Backdrop** options and some **Rectangle** elements.



Step 14

You can select one of the options of *Acrylic* or *Mica* to see the **System Backdrop** applied to the **Window** or you can select *None* to clear the **System Backdrop**.



Step 15

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!

