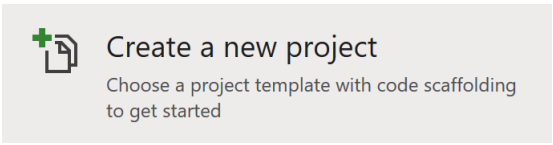
Universal Windows Platform – Connected Animation

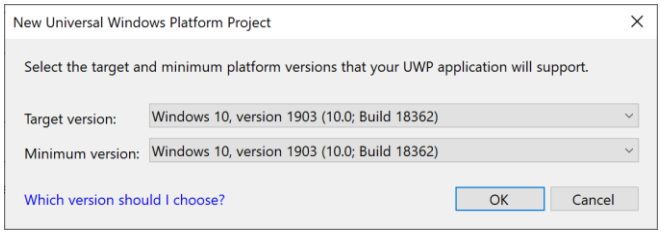
Connected Animation shows how to use a **Connected Animation** which is part of the **Fluent Design System** in **Windows 10**

Step 1



Follow **Setup and Start** on how to Install and/or Get Started with **Visual Studio 2019** if not already or in **Windows 10** choose **Start**, find and select **Visual Studio 2019** then from the **Get started** screen select **Create a new project**

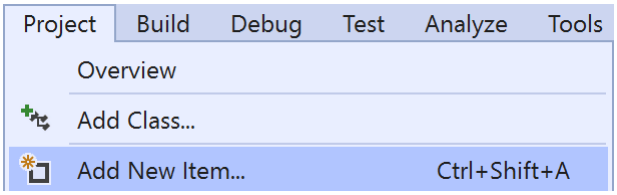
Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **ConnectedAnimation** and select **Create**



Finally, in **New Universal Windows Platform Project** pick the **Target version** and **Minimum version** to be at least **Windows 10, version 1903 (10.0; Build 18362)** and then select **OK**

Target Version will control the most recent features of Windows 10 your application can use. To make sure you always have the most recent version, check for any Notifications or Updates in Visual Studio 2019

Step 2



Choose **Project** then **Add New Item...** from the **Menu** in **Visual Studio 2019**

Step 3

Then choose **Code File** from **Add New Item** in **Visual Studio 2019**, enter the **Name** as **Library.cs** and select **Add**

Universal Windows Platform – Connected Animation

Step 4

In the **Code** View of **Library.cs** will be displayed and in this the following should be entered:

```
using System.Linq;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Media.Animation;
using Windows.UI.Xaml.Shapes;

public static class Library
{
    private const string animate_back = "AnimateBack";
    private const string animate_next = "AnimateNext";

    private static Windows.UI.Xaml.Media.Animation.
    ConnectedAnimation _animation;

    public static string Current { get; set; }


}
```

There is a `using` statement to include functionality and there are `const` of `string` and a `Windows.UI.Xaml.Media.Animation.ConnectedAnimation` and a `string` property

Then below the **`public static string Current { get; set; }`** line the following **`public static`** **methods** should be entered:

```
public static void Back(ref ListView listview)
{
    Rectangle rectangle = (Rectangle)listview.Items
    .SingleOrDefault(f => ((Rectangle)f).Tag.Equals(Current));
    _animation = ConnectedAnimationService.GetForCurrentView()
    .GetAnimation(animate_back);
    _animation?.TryStart(rectangle);
}

public static Brush Next(ref object selected)
{
    Rectangle rectangle = (Rectangle)selected;
    Current = (string)rectangle.Tag;
    _animation = ConnectedAnimationService.GetForCurrentView()
    .PrepareToAnimate(animate_next, rectangle);
    return rectangle.Fill;
}
```

The `Back(...)` method takes a `ListView` parameter and gets a `Rectangle` from the `ListView` and then gets the `Windows.UI.Xaml.Media.Animation.ConnectedAnimation` and calls the `TryStart` method on it. The `Next` method takes an `object` parameter which will be a `Rectangle` and then gets the `Windows.UI.Xaml.Media.Animation.ConnectedAnimation` for it and calls the `PrepareToAnimate` method of the `ConnectedAnimationService.GetForCurrentView`

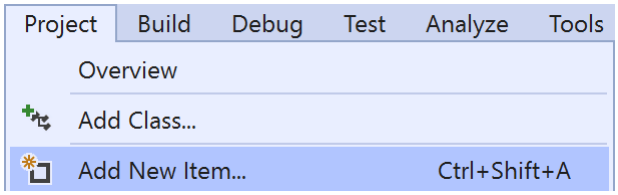
Universal Windows Platform – Connected Animation

Finally below the **`public static Brush Next(ref object selected) { ... }` method** the following **`public static` methods** should be entered:

```
public static void From(ref Rectangle from)
{
    _animation =
    ConnectedAnimationService.GetForCurrentView()
    .PrepareToAnimate(animate_back, from);
}

public static void Loaded(ref Rectangle rectangle)
{
    _animation =
    ConnectedAnimationService.GetForCurrentView()
    .GetAnimation(animate_next);
    rectangle.Opacity = 1;
    _animation?.TryStart(rectangle);
}
```

The `From(...)` method calls the `ConnectedAnimationService.GetForCurrentView` method of `PrepareToAnimate`. The `Loaded(...)` method takes a `Rectangle` parameter and this calls the `GetAnimation` method of `ConnectedAnimationService.GetForCurrentView` and will set the `Opacity` to `1` and calls the `TryStart` method

Step 5



Choose **Project** then **Add New Item...** from the **Menu** in **Visual Studio 2019**

Step 6

Then choose **Blank Page** from **Add New Item** in **Visual Studio 2019**, enter the **Name** as **DetailPage.xaml** and select **Add**

Step 7

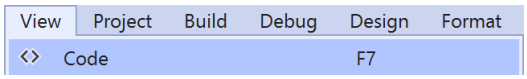
In the **Design** View and **XAML** View of **Visual Studio 2019** will be displayed, and in this between the **`Grid`** and **`/Grid`** elements enter the following **XAML**:

```
<Rectangle Margin="50" Name="Target" Opacity="0" Loaded="Target_Loaded"/>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Icon="Back" Label="Back" Click="Back_Click"/>
</CommandBar>
```

The first block of XAML is a Rectangle Control and the second block of XAML is a CommandBar with an AppBarButton for Back

Universal Windows Platform – Connected Animation

Step 8



Choose **View** then **Code** from the **Menu** in **Visual Studio 2019**

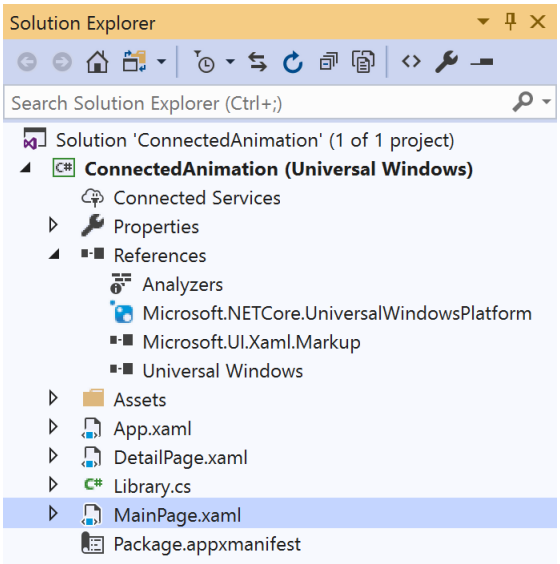
Step 9

Once in the **Code** View, below the end of **`public DetailPage() { ... }`** the following Code should be entered:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    Target.Fill = (SolidColorBrush)e.Parameter;
}

protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
{
    if (e.NavigationMode == NavigationMode.Back)
        Library.From(ref Target);
    base.OnNavigatingFrom(e);
}

private void Target_Loaded(object sender, RoutedEventArgs e)
{
    Library.Loaded(ref Target);
}

private void Back_Click(object sender, RoutedEventArgs e)
{
    this.Frame.GoBack();
}
```

`OnNavigatedTo` event handler will set the `Fill` property of the `Rectangle`, `OnNavigatingFrom` event handler will call the `From` method in the `Library` class and `Back_Click` will call `GoBack` to navigate to the previous XAML Page, MainPage.xaml

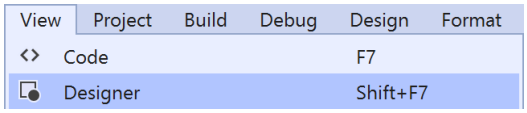
Step 10



In the **Solution Explorer** of **Visual Studio 2019** select **MainPage.xaml**

Universal Windows Platform – Connected Animation

Step 11



Choose **View** then **Designer** from the **Menu** in **Visual Studio 2019**

Step 12

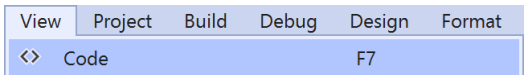
In the **Design** View and **XAML** View of **Visual Studio 2019** will be displayed, and in this between the **Grid** and **/Grid** elements enter the following **XAML**:

```
<ListView Name="Display" Margin="50">
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Black" Fill="Black" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Gray" Fill="Gray" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Red" Fill="Red" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Orange" Fill="Orange" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Yellow" Fill="Yellow" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Green" Fill="Green" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Cyan" Fill="Cyan" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Blue" Fill="Blue" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Magenta" Fill="Magenta" Tapped="Rectangle_Tapped"/>
    <Rectangle Margin="10" Width="64" Height="64"
    Tag="Purple" Fill="Purple" Tapped="Rectangle_Tapped"/>
</ListView>
```

The main block of XAML is a ListView which contains Rectangle Controls with their Tapped handler set properties set enabling the Control to support drag-and-drop. The second block of XAML is the CommandBar which contains the Add – to add to the ListBox and Remove - to remove items from the ListBox

Universal Windows Platform – Connected Animation

Step 13



Choose **View** then **Code** from the **Menu** in **Visual Studio 2019**

Step 14

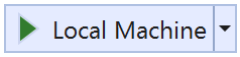
Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    if (e.NavigationMode == NavigationMode.Back)
        Library.Back(ref Display);
    base.OnNavigatedTo(e);
}

private void Rectangle_Tapped(object sender, TappedRoutedEventArgs e)
{
    this.Frame.Navigate(typeof(DetailPage), Library.Next(ref sender));
}
```

`OnNavigatedTo` event handler calls the `Back` method from the `Library` class and `Rectangle_Tapped` calls the `Navigate` method of the Page Frame and pass the `DetailPage` and the result of the `Next` method in the `Library` class

Universal Windows Platform – Connected Animation

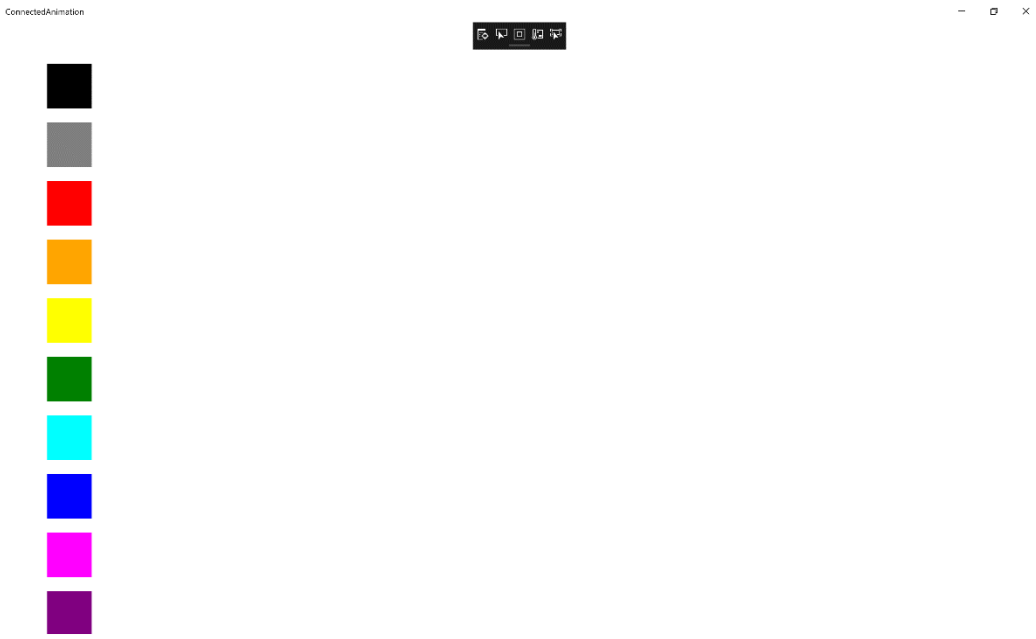
Step 10



That completes the **Universal Windows Platform** Application, in **Visual Studio 2019** select **Local Machine** to run the Application

Step 11

Once the Application is running you can tap on any of the **Rectangle** Controls, this will Navigate to the **DetailsPage** to show a larger version of a **Rectangle** with the same **Fill** but will use a **Connected Animation** to transition to and from that page



Step 12

To Exit the Application, select the **Close** button in the top right of the Application