

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

Windows App SDK

Light Effect





# Light Effect

**Light Effect** shows how you can use **PointLight** with an element to create a **Light Effect** in 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. | 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 *LightEffect*, then select a Location and then select **Create** to start a new **Solution**. | **Graphical user interface, text, application, email  Description automatically generated** |

## Step 2

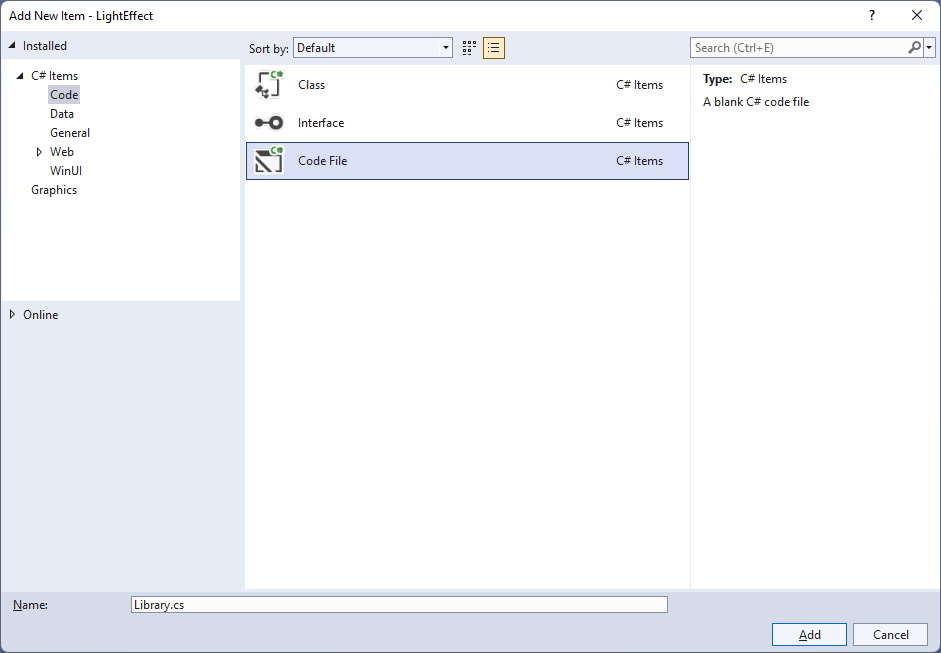
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

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



## 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.Composition;

using Microsoft.UI.Xaml;

using Microsoft.UI.Xaml.Hosting;

using System;

internal class Library

{

private PointLight \_light;

public void SetLight(FrameworkElement element)

{

var visual = ElementCompositionPreview.GetElementVisual(element);

var compositor = visual.Compositor;

\_light = compositor.CreatePointLight();

\_light.Color = Colors.White;

\_light.CoordinateSpace = visual;

\_light.Targets.Add(visual);

\_light.Offset = new System.Numerics.Vector3(

-(float)element.ActualWidth \* 2,

(float)element.ActualHeight / 2,

(float)element.ActualHeight);

var animation = compositor.CreateScalarKeyFrameAnimation();

animation.IterationBehavior = AnimationIterationBehavior.Forever;

animation.InsertKeyFrame(1, 2 \* (float)element.ActualWidth);

animation.Duration = TimeSpan.FromSeconds(5.0f);

\_light.StartAnimation("Offset.X", animation);

}

public void ClearLight()

{

if (\_light != null)

\_light.Targets.RemoveAll();

}

}

The **Class** that has been defined in *Library.cs* has a **Member** for a **PointLight** then there is a **Method** of **SetLight** which will create a **Light Effect** for a **FrameworkElement** by first creating an **Element Visual** with **ElementCompositionPreview** you’ll also notice the use of **var**, which means the type of the value doesn’t need to be explicitly specified, instead it will be **Inferred**. Then a **Compositor** is obtained from this a **PointLight** is configured where various values are set to display it as needed. There is also an **Animation** used which will make the **PointLight** move across the **FrameworkElement** with the specified **Behaviour**. The other **Method** of **ClearLight** is used to remove the **Light Effect** of the **PointLight**.

## 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">

<Rectangle Name="Display" Height="400" Width="400"

Margin="50" Stretch="Uniform" Fill="#FF5c2d91"/>

</Viewbox>

<CommandBar VerticalAlignment="Bottom" HorizontalAlignment="Stretch" >

<AppBarButton Icon="Accept" Label="Accept" Click="Accept\_Click"/>

<AppBarButton Icon="Cancel" Label="Clear" Click="Clear\_Click"/>

</CommandBar>

</Grid>

This **XAML** features a **Grid** with a **ViewBox** which is used to **Scale** elements, then within this is a **Rectangle** which is the **FrameworkElement** that will have the **Light Effect** applied to it. Then there is a **CommandBar** with an **AppBarButton** to apply the **Light Effect** of *Accept* and another to remove it of *Clear*.

## 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**. |  |

## 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 readonly Library \_library = new();

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

{

\_library.SetLight(Display);

}

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

{

\_library.ClearLight();

}

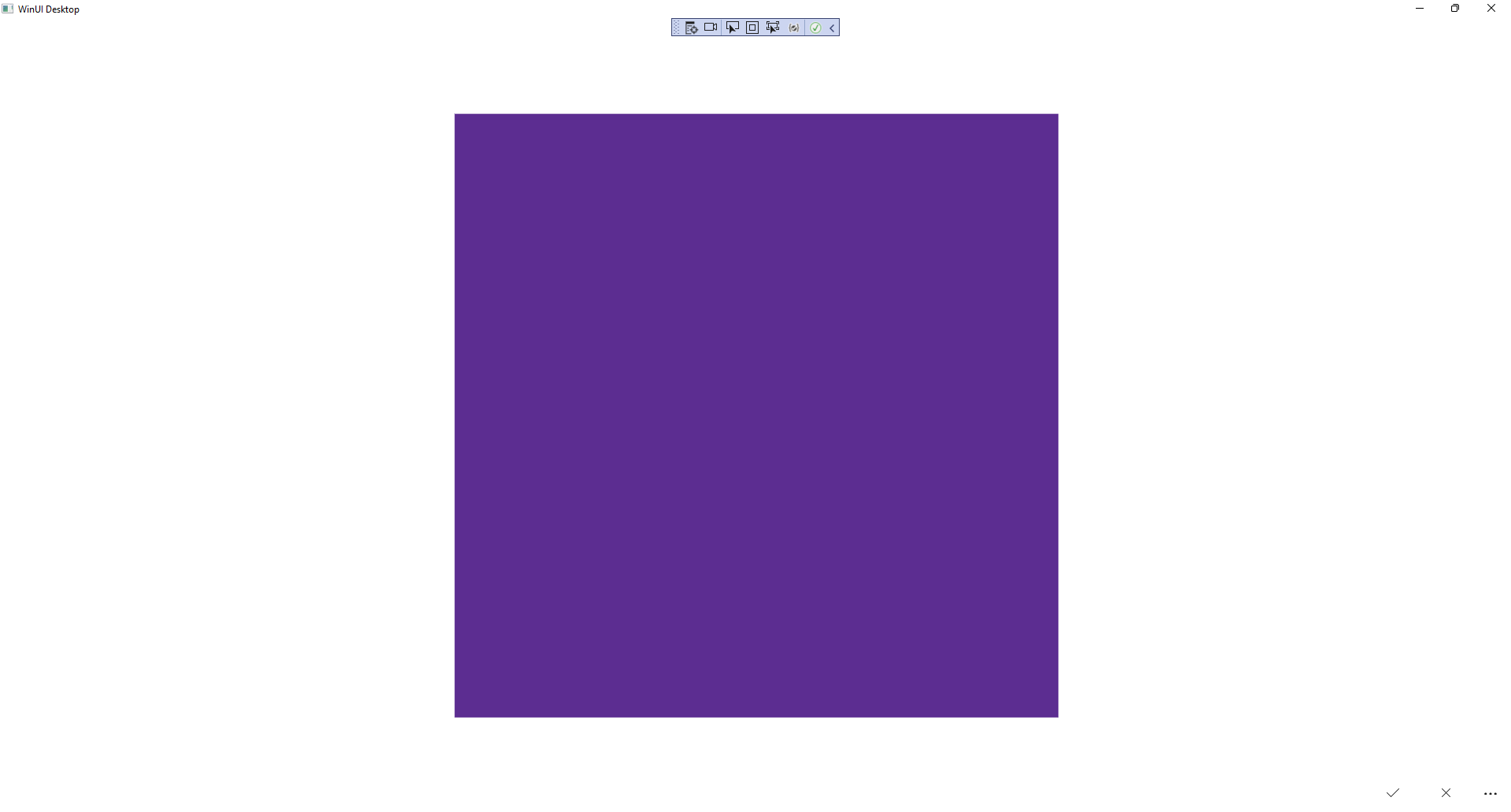
The **Method** of **Accept\_Click** will call the **Method** within *Library.cs* of **SetLight** from an **Instance** of **Library** called **\_library** created with **new()** and **Clear\_Click** will call the **Method** of **ClearLight**.

## Step 11

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

## Step 12

Once running you should see a **Rectangle** and **CommandBar** with the *Accept* and *Clear* options.

****

## Step 13

You can select *Apply* to set the **Light Effect** and *Clear* to remove the **Light Effect**



## 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](https://tutorialr.com)! |  |