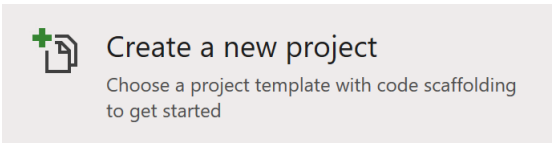
Universal Windows Platform – Photo Rotate

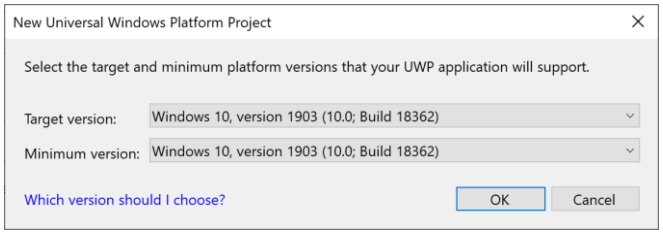
Photo Rotate shows how to use **BitmapTransform** and related **properties** and **methods** to create a simple image-rotating application

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**

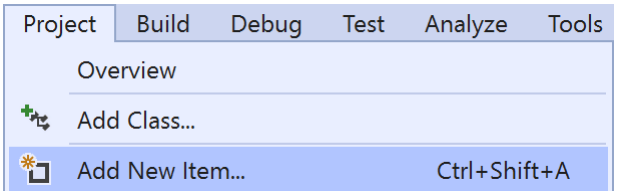
Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **PhotoRotate** and select **Create**



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 – Photo Rotate

Step 4

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

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using System.Threading.Tasks;
using Windows.Graphics.Imaging;
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media.Imaging;

public class Library
{
    private int _angle;
    private StorageFile _file;
    private WriteableBitmap _bitmap;

    private readonly Dictionary<int, BitmapRotation> rotation_angles =
        new Dictionary<int, BitmapRotation>()
    {
        { 0, BitmapRotation.None },
        { 90,  BitmapRotation.Clockwise90Degrees },
        { 180,  BitmapRotation.Clockwise180Degrees },
        { 270, BitmapRotation.Clockwise270Degrees },
        { 360, BitmapRotation.None }
    };

    private const string file_extension = ".jpg";


}
```

There are `using` statements to include necessary functionality. Also, there are an `int` to store the rotation angle, then a `StorageFile` to get or set the File of the photo and a `Dictionary` to store the supported rotation angles

Then below the **`private const string file_extension = ".jpg";`** line the following **method** should be entered:

```
private async Task<WriteableBitmap> ReadAsync()
{
    using (IRandomAccessStream stream = await
    _file.OpenAsync(FileAccessMode.ReadWrite))
    {
        BitmapDecoder decoder = await BitmapDecoder
        .CreateAsync(BitmapDecoder.JpegDecoderId, stream);
        uint width = decoder.PixelWidth;
        uint height = decoder.PixelHeight;
        if (_angle % 180 != 0)
        {
            width = decoder.PixelHeight;
            height = decoder.PixelWidth;
        }
        BitmapTransform transform = new BitmapTransform
        {
            Rotation = rotation_angles[_angle]
        };
        PixelDataProvider data = await decoder.GetPixelDataAsync(
        BitmapPixelFormat.Bgra8, BitmapAlphaMode.Ignore, transform,
        ExifOrientationMode.IgnoreExifOrientation,
        ColorManagementMode.DoNotColorManage);
        _bitmap = new WriteableBitmap((int)width, (int)height);
        using (Stream pixels = _bitmap.PixelBuffer.AsStream())
        {
            pixels.Write(data.DetachPixelData(), 0, (int)pixels.Length);
        }
    }
    return _bitmap;
}
```

`ReadAsync()` is used to get an `IRandomAccessStream` from the `StorageFile` and get the image with a `BitmapDecoder`. The photo is then manipulated to introduce a transformation – in this case the rotation with the `PixelDataProvider` and this information is then written back to the File to produce the rotated image

Universal Windows Platform – Photo Rotate

Next below the **`private void ReadAsync() { ... }` method** the following **method** should be entered:

```
private async void WriteAsync()
{
    using (IRandomAccessStream stream = await
    _file.OpenAsync(FileAccessMode.ReadWrite))
    {
        BitmapEncoder encoder = await BitmapEncoder.CreateAsync
        (BitmapEncoder.JpegEncoderId, stream);
        encoder.SetPixelData(BitmapPixelFormat.Bgra8,
        BitmapAlphaMode.Ignore,
        (uint)_bitmap.PixelWidth, (uint)_bitmap.PixelHeight,
        96.0, 96.0, _bitmap.PixelBuffer.ToArray());
        await encoder.FlushAsync();
    }
}
```

`WriteAsync()` is used encode any resulting image as the correct format with `BitmapEncoder` set to the settings optimised for a photo

Then below the **`private void WriteAsync() { ... }` method** the following `public` **method** should be entered:

```
public async void OpenAsync(Image display)
{
    _angle = 0;
    try
    {
        FileOpenPicker picker = new FileOpenPicker
        {
            SuggestedStartLocation = PickerLocationId.PicturesLibrary
        };
        picker.FileTypeFilter.Add(file_extension);
        _file = await picker.PickSingleFileAsync();
        if (_file != null)
        {
            display.Source = await ReadAsync();
        }
    }
    catch
    {

    }
}
```

`OpenAsync` is used to get a photo with a `FileOpenPicker` and calls the `ReadAsync` method* to get the file and set the `Source` of the `Image` passed in

Universal Windows Platform – Photo Rotate

Next below the **`private void OpenAsync(...) { ... }` method** the following `public` **method** should be entered:

```
public async void SaveAsync()
{
    try
    {
        FileSavePicker picker = new FileSavePicker
        {
            DefaultFileExtension = file_extension,
            SuggestedFileName = "Picture",
            SuggestedStartLocation = PickerLocationId.PicturesLibrary
        };
        picker.FileTypeChoices.Add("Picture",
        new List<string>() { file_extension });
        _file = await picker.PickSaveFileAsync();
        if (_file != null)
        {
            WriteAsync();
        }
    }
    catch
    {

    }
}
```

`SaveAsync` is used to store a photo after it has be rotated with the `FileSavePicker` used along with the `WriteAsync` Method

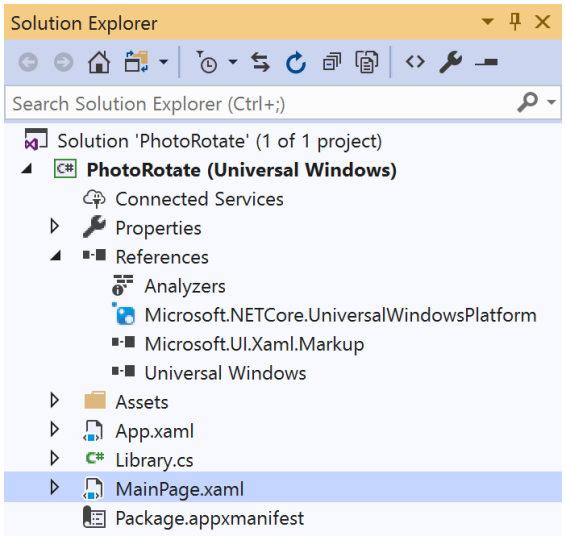
Finally after **`private void SaveAsync() { ... }` method** the following `public` **method** should be entered:

```
public async void RotateAsync(Image display)
{
    if (_angle == 360) _angle = 0;
    _angle += 90;
    display.Source = await ReadAsync();
}
```

`RotateAsync` is used to rotate the image and increments the value to be used by 90 degrees each time it is called up to 360 degrees when it is reset, the image is then obtained again with this rotation using the `ReadAsync` Method

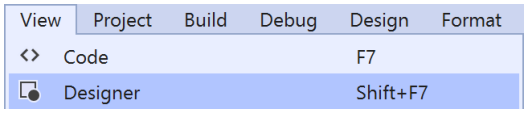
Universal Windows Platform – Photo Rotate

Step 5



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

Step 6



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

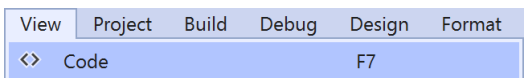
Step 7

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**:

```
<ScrollViewer VerticalScrollBarVisibility="Auto"
HorizontalScrollBarVisibility="Auto" ZoomMode="Enabled">
    <Image Name="Display"/>
</ScrollViewer>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Icon="OpenFile" Label="Open" Click="Open_Click"/>
    <AppBarButton Icon="Save" Label="Save" Click="Save_Click"/>
    <AppBarButton Icon="Rotate" Label="Rotate" Click="Rotate_Click"/>
</CommandBar>
```

The first block of XAML the main user interface and features a ScrollViewer containing an Image Control that can be zoomed in or out of so you can view the image at any needed size with pinch-to-zoom or scroll-wheel. The second block of XAML is is the CommandBar which contains Open – to read and show a photo in the Image Control and Save to write a photo after it has been rotated and finally Rotate to perform the rotation

Step 8



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

Universal Windows Platform – Photo Rotate

Step 9

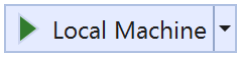
Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
Library library = new Library();

private void Open_Click(object sender, RoutedEventArgs e)
{
    library.OpenAsync(Display);
}

private void Save_Click(object sender, RoutedEventArgs e)
{
    library.SaveAsync();
}

private void Rotate_Click(object sender, RoutedEventArgs e)
{
    library.RotateAsync(Display);
}
```

Below the MainPage(...) method an instance of the `Library` Class is created. `Open_Click(...)` event handler is used to call the `OpenAsync` method to get a photo. `Save_Click(...)` calls `SaveAsync` which is used to store a photo and `Rotate_Click(...)` calls `RotateAsync` and is used to perform the rotation of the photo

Universal Windows Platform – Photo Rotate

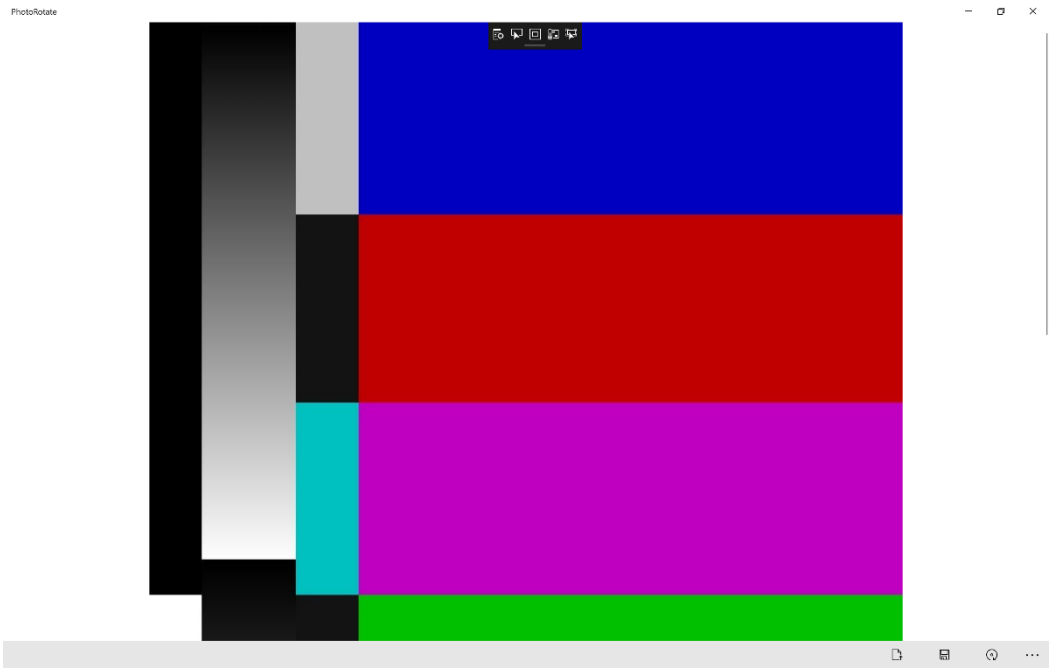
Step 10



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

Step 11

Once the Application running you can use **Open** to select a Photo to **Rotate** after which you can then **Save** the rotated photo



Step 12

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