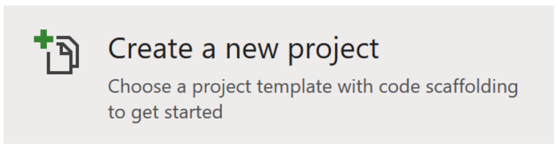
# Universal Windows Platform – Lucky Roulette

**Lucky Roulette** shows how to create a **Roulette** game using **Grid** and **Ellipse** controls and see what number will be randomly selected

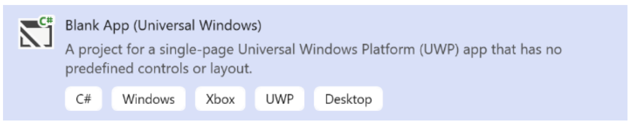
## Step 1



Create a new project

Choose a project template with code scaffolding to get started

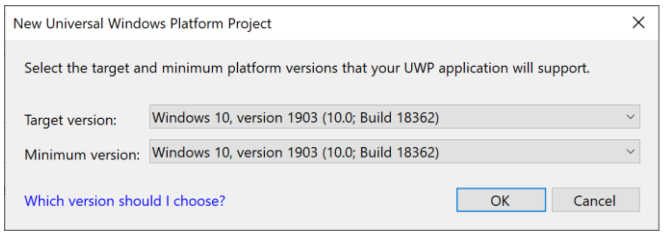
Follow **Setup and Start** on how to Install and/or Get Started with **Visual Studio 2019** if not already or in **Windows 10** choose **Start**, find and select **Visual Studio 2019** then from the **Get started** screen select **Create a new project**



Blank App (Universal Windows)

A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.

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



New Universal Windows Platform Project

Select the target and minimum platform versions that your UWP application will support.

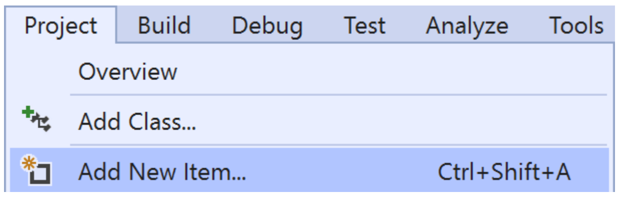
Target version: Windows 10, version 1903 (10.0; Build 18362)

Minimum version: Windows 10, version 1903 (10.0; Build 18362)

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**

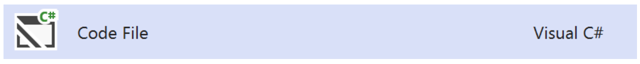
Target Version will control the most recent features of Windows 10 your application can use. To make sure you always have the most recent version, check for any Notifications or Updates in Visual Studio 2019

## 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 – Lucky Roulette

## 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.Linq;
using Windows.Foundation;
using Windows.UI;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Shapes;

public class Library
{
    private const string title = "Lucky Roulette";
    private const int size = 400;
    private const int rim = 50;

    private readonly int[] wheel =
    {
        0, 32, 15, 19, 4, 21, 2, 25, 17,
        34, 6, 27, 13, 36, 11, 30, 8, 23,
        10, 5, 24, 16, 33, 1, 20, 14, 31,
        9, 22, 18, 29, 7, 28, 12, 35, 3, 26
    };
    private readonly List<int> _values = Enumerable.Range(0, 36).ToList();

    private int _spins = 0;
    private int _spinValue = 0;
    private int _pickValue = 0;
    private Random _random = new Random((int)DateTime.Now.Ticks);


}
```

There are `using` statements to include necessary functionality. Also, there are `private const` for the setup of the game and for the values that will represent the `size` and values that will appear on the `wheel`. There is also a `List<int>` that will be used to be selected from.

Then below the **`private string[,] _board = new string[size, size];`** line the following **methods** should be entered:

```
private void Show(string content, string title)
{
    _ = new MessageDialog(content, title).ShowAsync();
}

private bool IsOdd(int value)
{
    return value % 2 != 0;
}

private Color Fill(int value)
{
    Color fill;
    if (value >= 1 && value <= 10 ||
        value >= 19 && value <= 28)
    {
        fill = IsOdd(value) ? Colors.Black : Colors.DarkRed;
    }
    else if (value >= 11 && value <= 18 ||
                value >= 29 && value <= 36)
    {
        fill = IsOdd(value) ? Colors.DarkRed : Colors.Black;
    }
    else if (value == 0)
    {
        fill = Colors.DarkGreen;
    }
    return fill;
}
```

The `Show` method is used to display a basic `MessageDialog`, `IsOdd` method is used to work out if a number is odd or even and is used by the `Fill` method to get the `Color` that should be used

# Universal Windows Platform – Lucky Roulette

Next below the `private Color Fill(int value) { ...}` **method** the following **method** should be entered:

```
private Grid Pocket(int value)
{
    Color fill = Fill(value);
    Grid grid = new Grid()
    {
        Width = size,
        Height = size
    };
    Grid pocket = new Grid()
    {
        Width = 26,
        Height = rim,
        CornerRadius = new CornerRadius(4),
        Background = new SolidColorBrush(fill),
        VerticalAlignment = VerticalAlignment.Top
    };
    TextBlock text = new TextBlock()
    {
        FontSize = 20,
        Text = $"{value}",
        VerticalAlignment = VerticalAlignment.Top,
        Foreground = new SolidColorBrush(Colors.Gold),
        HorizontalAlignment = HorizontalAlignment.Center
    };
    Ellipse ball = new Ellipse()
    {
        Width = 20,
        Height = 20,
        Opacity = 0,
        Name = $"{value}",
        Margin = new Thickness(0, 0, 0, 4),
        Fill = new SolidColorBrush(Colors.Snow),
        VerticalAlignment = VerticalAlignment.Bottom
    };
    pocket.Children.Add(text);
    pocket.Children.Add(ball);
    grid.Children.Add(pocket);
    return grid;
}
```

The `Pocket` method is used to create the elements that will make up the Roulette Wheel and uses the `Fill` method for the background colour of each element

Then after the **`private Grid Pocket(int value) { ... }`** **method** the following **method** should be entered:

```
private Viewbox Layout()
{
    Canvas canvas = new Canvas()
    {
        Width = size,
        Height = size,
    };
    Ellipse ellipse = new Ellipse()
    {
        Width = size,
        Height = size,
        StrokeThickness = rim,
        Stroke = new SolidColorBrush(Colors.Peru)
    };
    canvas.Children.Add(ellipse);
    int index = 0;
    double radiusX = canvas.Width * 0.5;
    double radiusY = canvas.Height * 0.5;
    double delta = 2 * Math.PI / wheel.Length;
    Point centre = new Point(canvas.Width / 2, canvas.Height / 2);
    foreach (int value in wheel)
    {
        Grid pocket = Pocket(value);
        Size size = new Size(pocket.DesiredSize.Width,
            pocket.DesiredSize.Height);
        double angle = index * delta;
        double x = centre.X + radiusX *
            Math.Cos(angle) - size.Width / 2;
        double y = centre.Y + radiusY *
            Math.Sin(angle) - size.Height / 2;
        pocket.RenderTransformOrigin = new Point(0.5, 0.5);
        pocket.RenderTransform = new RotateTransform()
        {
            Angle = angle * 180 / Math.PI
        };
        pocket.Arrange(new Rect(x, y, size.Width, size.Height));
        canvas.Children.Add(pocket);
        index++;
    }
    Viewbox viewbox = new Viewbox()
    {
        Child = canvas
    };
    return viewbox;
}
```

The `Layout` method is used to create the look-and-feel of the Roulette Wheel made up of `Pocket` elements which are positioned around a `Canvas`

# Universal Windows Platform – Lucky Roulette

Next after the **`private Viewbox Layout()` method** the following **method** should be entered:

```
private void Set(Grid grid, int value, byte opacity)
{
    UIElement element = (UIElement)grid.FindName($"{value}");
    if (element != null) element.Opacity = opacity;
}
```

The `Set` method is used to set the `Opacity` of a `UIElement`

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

```
private async void Choose(Grid grid)
{
    for (int i = 0; i < _values.Count; i++)
    {
        Set(grid, _values[i], 0);
    }
    ComboBox combobox = new ComboBox()
    {
        SelectedIndex = 0,
        ItemsSource = _values,
        Margin = new Thickness(10),
        HorizontalAlignment = HorizontalAlignment.Center
    };
    ContentDialog dialog = new ContentDialog()
    {
        Content = combobox,
        Title = "Pick a Number",
        PrimaryButtonText = "Spin",
        SecondaryButtonText = "Cancel"
    };
    ContentDialogResult result = await dialog.ShowAsync();
    if (result == ContentDialogResult.Primary)
    {
        _spins++;
        _spinValue = _values[_random.Next(0, _values.Count)];
        _pickValue = (int)((ComboBox)dialog.Content).SelectedValue;
        Set(grid, _spinValue, 1); // Show Ball
        if (_spinValue == _pickValue) // Check Win
        {
            _spins = 0;
            Show($"Won {_spins} with {_spinValue}", title);
        }
        else
        {
            Show($"Lost {_spins} with {_pickValue} was {_spinValue}", title);
        }
    }
}
```

The `Choose` method a `Grid` and sets up the layout of the game using the `Add(...)` method and `_board`

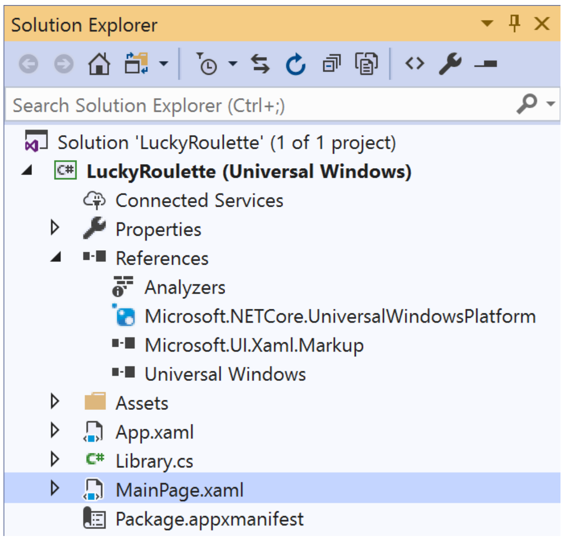
# Universal Windows Platform – Lucky Roulette

Finally after the **`private async void Choose(Grid grid) { ... }` method** the following **`public` methods** should be entered:

```
public void New(Grid grid)
{
    _spins = 0;
    grid.Children.Clear();
    grid.Children.Add(Layout());
}

public void Play(Grid grid)
{
    if (!grid.Children.Any()) New(grid);
    Choose(grid);
}
```

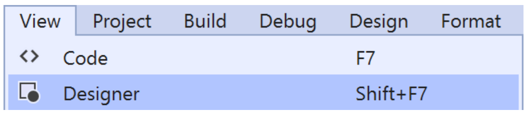
The `New` method will setup the layout of the `Grid` using the `Layout` method and the `Play` method will call `New` method if not already done then will call the `Choose` method to play the game

## 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**

# Universal Windows Platform – Lucky Roulette

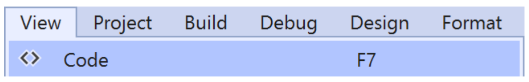
## Step 7

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**:

```
<Viewbox>
    <Grid Margin="50" Name="Display"
    HorizontalAlignment="Center"
    VerticalAlignment="Center"/>
</Viewbox>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Icon="Page2" Label="New" Click="New_Click"/>
    <AppBarButton Icon="Play" Label="Play" Click="Play_Click"/>
</CommandBar>
```

The first block of XAML the main user interface is a Viewbox to contain a Grid which will display the game. The second block of XAML is the CommandBar which contains New to setup the game and Play to start playing

## Step 8



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

## Step 9

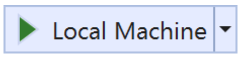
Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
Library library = new Library();

private void New_Click(object sender, RoutedEventArgs e)
{
    library.New(Display);
}

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

Below the MainPage method an instance of the `Library` Class is created. The `New_Click` event handler will call the `New` method in the `Library` class and the `Play_Click` event handler will call the `Play` method

# Universal Windows Platform – Lucky Roulette

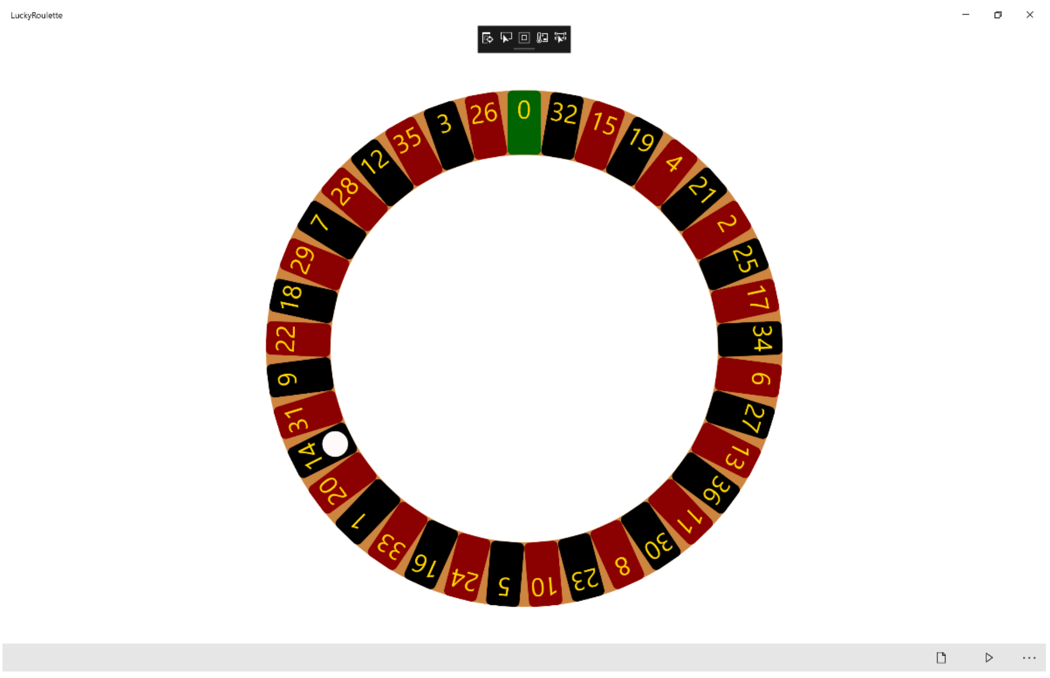
## Step 10



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 click **New** to setup the **Roulette Wheel** then click **Play** which will show a dialog where you can select a number from a dropdown list and choose **Spin** – if you get it right, you win and if not, you lose



## Step 12

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