tutorialr.com

tutorial

Windows App SDK

Lucky Wheel

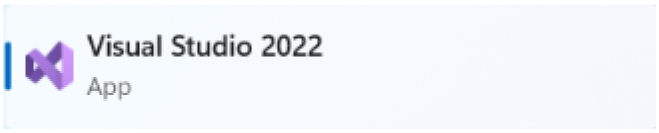
Lucky Wheel

Lucky Wheel shows how you can create spinning wheel game with a random chance of getting a particular score or you could lose it all, using a control in a toolkit from **NuGet** using the **Windows App SDK**.

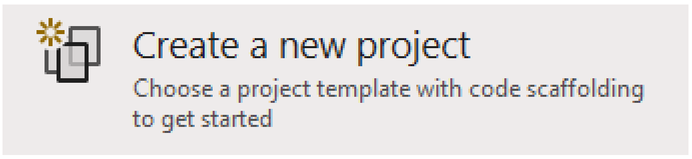
Step 1

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

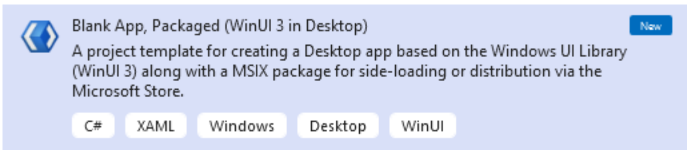
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



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


Create a new project
Choose a project template with code scaffolding to get started

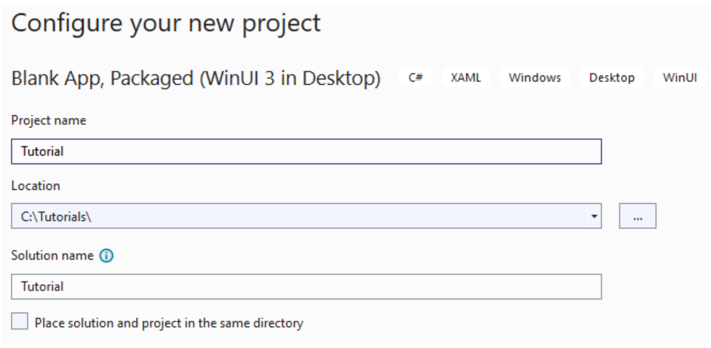
Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.



Blank App, Packaged (WinUI 3 in Desktop)

A project template for creating a Desktop app based on the Windows UI Library (WinUI 3) along with a MSIX package for side-loading or distribution via the Microsoft Store.

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



Configure your new project

Blank App, Packaged (WinUI 3 in Desktop)

Project name: Tutorial

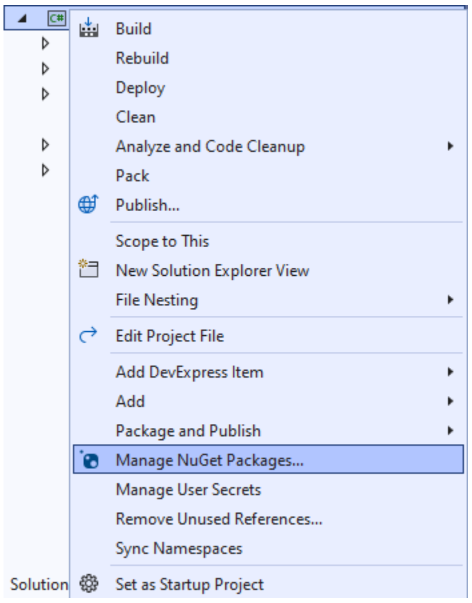
Location: C:\Tutorials\

Solution name: Tutorial

Place solution and project in the same directory

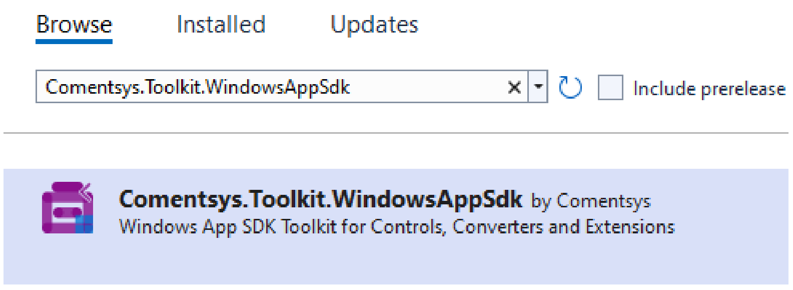
Step 2

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Manage NuGet Packages...**



Step 3

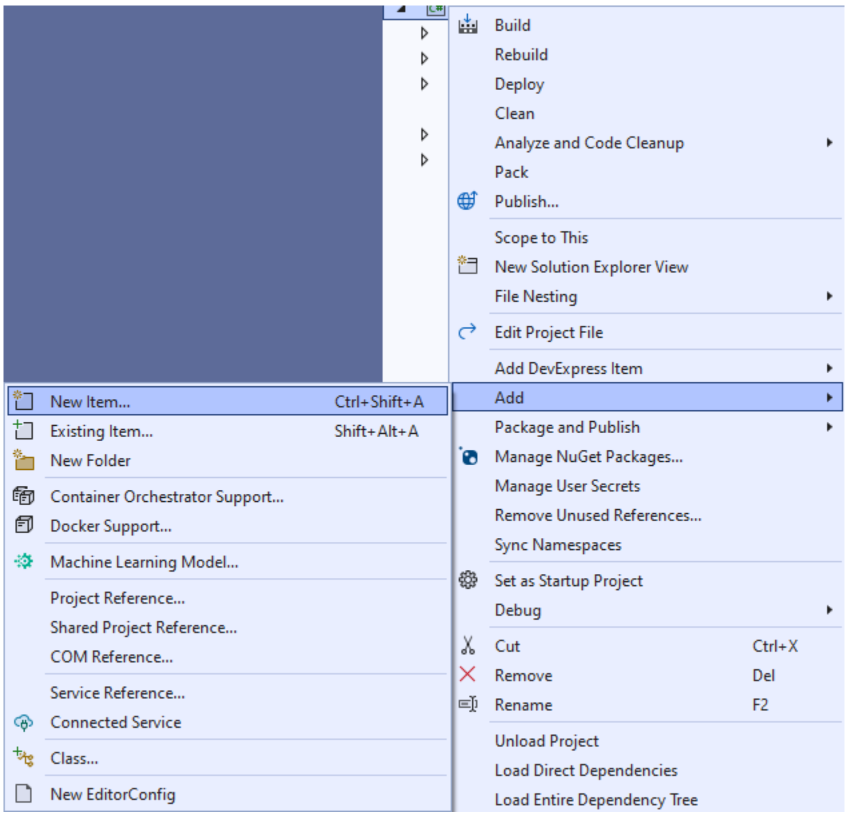
Then in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Toolkit.WindowsAppSdk** and then select **Comentsys.Toolkit.WindowsAppSdk by Comentsys** as indicated and select **Install**


Browse
Installed
Updates
Comentsys.Toolkit.WindowsAppSdk
Include prerelease
Comentsys.Toolkit.WindowsAppSdk by Comentsys
Windows App SDK Toolkit for Controls, Converters and Extensions

This will add the package for **Comentsys.Toolkit.WindowsAppSdk** to your **Project**. If you get the **Preview Changes** screen saying **Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.** You can read the message and then select **OK** to **Install** the package,, then you can close the **tab** for **Nuget: LuckyWheel** by selecting the **x** next to it.

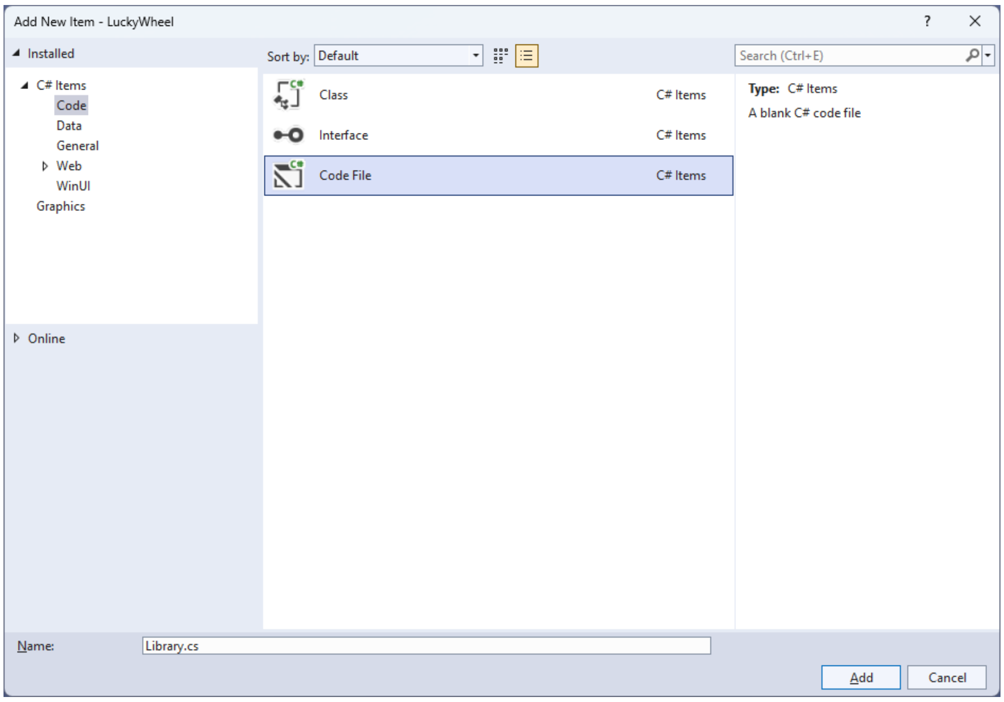
Step 4

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 5

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**.



Type: C# Items
A blank C# code file

Name: Library.cs

Step 6

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

```
using Comentsys.Toolkit.WindowsAppSdk;
using Microsoft.UI;
using Microsoft.UI.Text;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Documents;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Media.Animation;
using Microsoft.UI.Xaml.Shapes;
using System;
using System.Collections.Generic;
using System.Linq;
using Windows.Foundation;
using Windows.UI;

public class Library
{
    private const string title = "Lucky Wheel";
    private const int size = 400, hole = 60, radius = 200, circle = 360;
    private const int border = 4, marker = 30, duration = 5;
    private static readonly List<(string Value, Color Fill)> wedges = new()
    {
        ("1000", Colors.WhiteSmoke), ("600", Colors.LightGreen),
        ("500", Colors.Yellow), ("300", Colors.Red),
        ("500", Colors.Azure), ("800", Colors.Orange),
        ("550", Colors.Violet), ("400", Colors.Yellow),
        ("300", Colors.Pink), ("900", Colors.Red),
        ("500", Colors.Azure), ("300", Colors.LightGreen),
        ("900", Colors.Pink), ("LOSE", Colors.Black),
        ("600", Colors.Violet), ("400", Colors.Yellow),
        ("300", Colors.Azure), ("LOSE", Colors.Black),
        ("800", Colors.Red), ("350", Colors.Violet),
        ("450", Colors.Pink), ("700", Colors.LightGreen),
        ("300", Colors.Orange), ("600", Colors.Violet),
    };
    private static readonly double section = circle / wedges.Count;
    private readonly Random _random = new((int)DateTime.UtcNow.Ticks);

    // Variables, Get Ellipse, Add Circle, Get Sector & Add Section

    // Get Text & Add Text

    // Get Marker & Add Rotate

    // Set Rotate, Play & Add Wheel

    // Layout, Reset & New

}
```

`Class` defined so far *Library.cs* has **`using`** for package of **`Comentsys.Toolkit.WindowsAppSdk`** and others.

Step 7

Still in the **Class** for *Library.cs* after the **Comment** of **`// Variables, Get Ellipse, Add Circle, Get Sector & Add Section`** type the following **Variables** and **Methods** for **`GetEllipse`** which will get an **`Ellipse`** used by **`AddCircle`** along with **`GetSector`** which will get a **`Sector`** control used by **`AddSection`**.

```
private int _total;
private bool _over;
private bool _spin;
private double _position;
private double _selected;
private Canvas _canvas;
private Dialog _dialog;
private Storyboard _storyboard;

private Ellipse GetEllipse(double diameter, Color fill) => new()
{
    Width = diameter,
    Height = diameter,
    StrokeThickness = border,
    Fill = new SolidColorBrush(fill),
    Stroke = new SolidColorBrush(Colors.Gold)
};

private void AddCircle(Canvas canvas, double diameter)
{
    var circle = GetEllipse(diameter, Colors.Green);
    Canvas.SetLeft(circle, (size - diameter) / 2);
    Canvas.SetTop(circle, (size - diameter) / 2);
    canvas.Children.Add(circle);
}

private Sector GetSector(double start, double finish, double radius, Color fill)
{
    Sector sector = new()
    {
        Hole = hole,
        Start = start,
        Finish = finish,
        Radius = radius,
        Fill = new SolidColorBrush(fill)
    };
    Canvas.SetLeft(sector, (size - radius * 2) / 2);
    Canvas.SetTop(sector, (size - radius * 2) / 2);
    return sector;
}

private void AddSection(Canvas canvas, int index, double start)
{
    var finish = section;
    var sector = GetSector(start, finish, radius, wedges[index].Fill);
    canvas.Children.Add(sector);
}
```

Step 8

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Get Text & Add Text`** type in the following **Methods** for **`GetText`** which will create a **`TextBlock`** for amounts to be used by **`AddText`** which will position the amounts on the wheel.

```
private TextBlock GetText(string value, Color foreground)
{
    TextBlock text = new()
    {
        FontSize = 20,
        Margin = new Thickness(2),
        FontWeight = FontWeights.SemiBold,
        TextAlignment = TextAlignment.Center,
        Foreground = new SolidColorBrush(foreground)
    };
    for (int index = 0; index < value.Length; index++)
    {
        text.Inlines.Add(new Run()
        {
            Text = value[index] + Environment.NewLine
        });
    }
    text.Measure(new Size(
        double.PositiveInfinity,
        double.PositiveInfinity));
    return text;
}

private void AddText(Canvas canvas, int index, double start)
{
    double top = 0;
    var (value, fill) = wedges[index];
    var foreground = fill == Colors.Black ? Colors.White : Colors.Black;
    var text = GetText(value, foreground);
    double middle = text.DesiredSize.Width / 2;
    double left = (size / 2) - middle;
    Grid grid = new()
    {
        Height = radius,
        RenderTransform = new RotateTransform()
        {
            Angle = start,
            CenterX = middle,
            CenterY = radius
        }
    };
    grid.Children.Add(text);
    Canvas.SetLeft(grid, left);
    Canvas.SetTop(grid, top);
    canvas.Children.Add(grid);
}
```

Step 9

While still in the **Class** for *Library.cs* after the **Comment** of **`// Get Marker & Add Rotate`** type in the following **Methods** for **`GetMarker`** for the marker on top of the wheel and **`AddRotate`** for rotating it.

```
private Polygon GetMarker() => new()
{
    Width = marker,
    Height = marker / 2,
    Fill = new SolidColorBrush(Colors.Gold),
    VerticalAlignment = VerticalAlignment.Center,
    HorizontalAlignment = HorizontalAlignment.Center,
    Points =
    {
        new Point(0, 0),
        new Point(marker, 0),
        new Point(marker / 2, marker / 2)
    }
};

private void AddRotate()
{
    DoubleAnimation animation = new()
    {
        From = _position,
        To = circle * 2,
        EasingFunction = new QuadraticEase(),
        RepeatBehavior = new RepeatBehavior(1),
        Duration = new Duration(TimeSpan.FromSeconds(duration))
    };
    Storyboard.SetTargetProperty(animation,
        "(Canvas.RenderTransform).(RotateTransform.Angle)");
    Storyboard.SetTarget(animation, _canvas);
    _storyboard = new Storyboard();
    _storyboard.Completed += (object sender, object e) =>
    {
        _spin = false;
        var angle = circle - _selected - (section / 2);
        var index = (int)Math.Ceiling(angle / section);
        var (value, _) = wedges[index];
        if (int.TryParse(value, out int result) && !_over)
        {
            _total += result;
            _dialog.Show($"You Won {result}, Total is {_total}");
        }
        else
        {
            _over = true;
            _dialog.Show($"You Lose, Total was {_total}!");
        }
    };
    _storyboard.Children.Add(animation);
}
```

Step 10

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Set Rotate, Play & Add Wheel`** type in the following **Methods**:

```
private void SetRotate(double angle)
{
    var animation = _storyboard.Children.First() as DoubleAnimation;
    animation.From = _position;
    animation.To = circle * 2 + angle;
    _storyboard.Begin();
}

private void Play()
{
    if (!_spin)
    {
        _spin = true;
        if (_over)
        {
            _dialog.Show($"You Lost, Total was {_total}, Starting New Game");
            Reset();
        }
        else
        {
            _position = _selected;
            _selected = _random.Next(1, circle);
            SetRotate(_selected);
        }
    }
}

private void AddWheel(Canvas canvas, double diameter)
{
    var wheel = GetEllipse(diameter, Colors.Transparent);
    wheel.Tapped += (object sender, TappedRoutedEventArgs e) =>
        Play();
    Canvas.SetLeft(wheel, (size - diameter) / 2);
    Canvas.SetTop(wheel, (size - diameter) / 2);
    canvas.Children.Add(wheel);
}
```

`SetRotate` will trigger the animation to rotate or spin the wheel, **`Play`** will be used when spinning the wheel and will select the next value or if the game is over it will show a message and **`AddWheel`** which will add a transparent **`Ellipse`** which will capture **Events** when it is **`Tapped`** to trigger **`Play`**.

Step 11

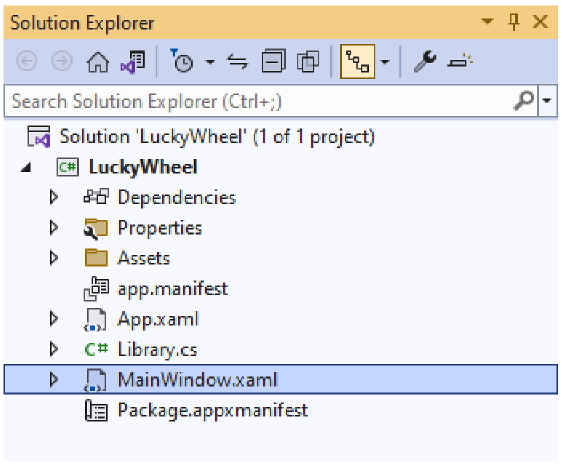
While still in the **Class** for *Library.cs* after the **Comment** of **// Layout, Reset & New** type in the following **Methods**:

```
private void Layout(Grid grid)
{
    grid.Children.Clear();
    StackPanel panel = new();
    _canvas = new Canvas()
    {
        Width = size,
        Height = size,
        RenderTransform = new RotateTransform()
        {
            Angle = 0,
            CenterX = radius,
            CenterY = radius
        }
    };
    var start = -(section / 2);
    for (int index = 0; index < wedges.Count; index++)
    {
        AddSection(_canvas, index, start);
        AddText(_canvas, index, start + (section / 2));
        start += section;
    }
    AddCircle(_canvas, hole * 2);
    AddWheel(_canvas, size + border);
    AddRotate();
    panel.Children.Add(GetMarker());
    panel.Children.Add(_canvas);
    grid.Children.Add(panel);
}

private void Reset()
{
    _total = 0;
    _spin = false;
    _over = false;
    _selected = 0;
}

public void New(Grid grid)
{
    Reset();
    Layout(grid);
    _dialog = new Dialog(grid.XamlRoot, title);
}
```

Layout will create the look-and-feel of the wheel by setting up all the elements, **Reset** will reset the values used in the game when a game is over or started and **New** will start a new game.

Step 12

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



Step 13

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 14

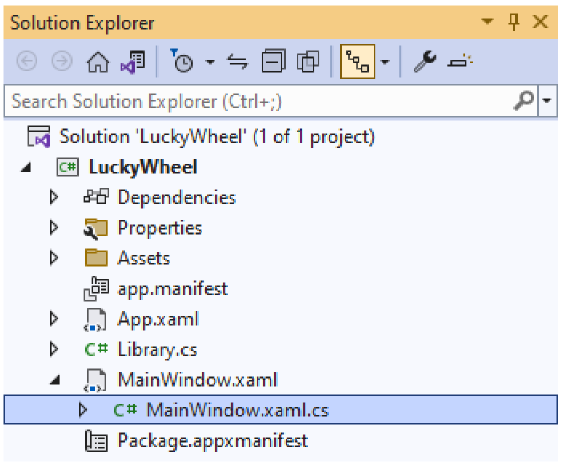
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

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

This **XAML** contains a **`Grid`** with a **`Viewbox`** which will scale a **`Grid`**. It has a **`Loaded`** event handler for **`New`** which is also shared by the **`AppBarButton`**.

Step 15

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 16

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 17

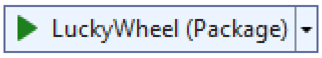
Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

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

private void New(object sender, RoutedEventArgs e) =>
    _library.New(Display);
```

Here an **Instance** of the **Class** of **`Library`** is created then below this is the **Method** of **`New`** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **`=>`** for an Expression Body which is useful when a **Method** only has one line.

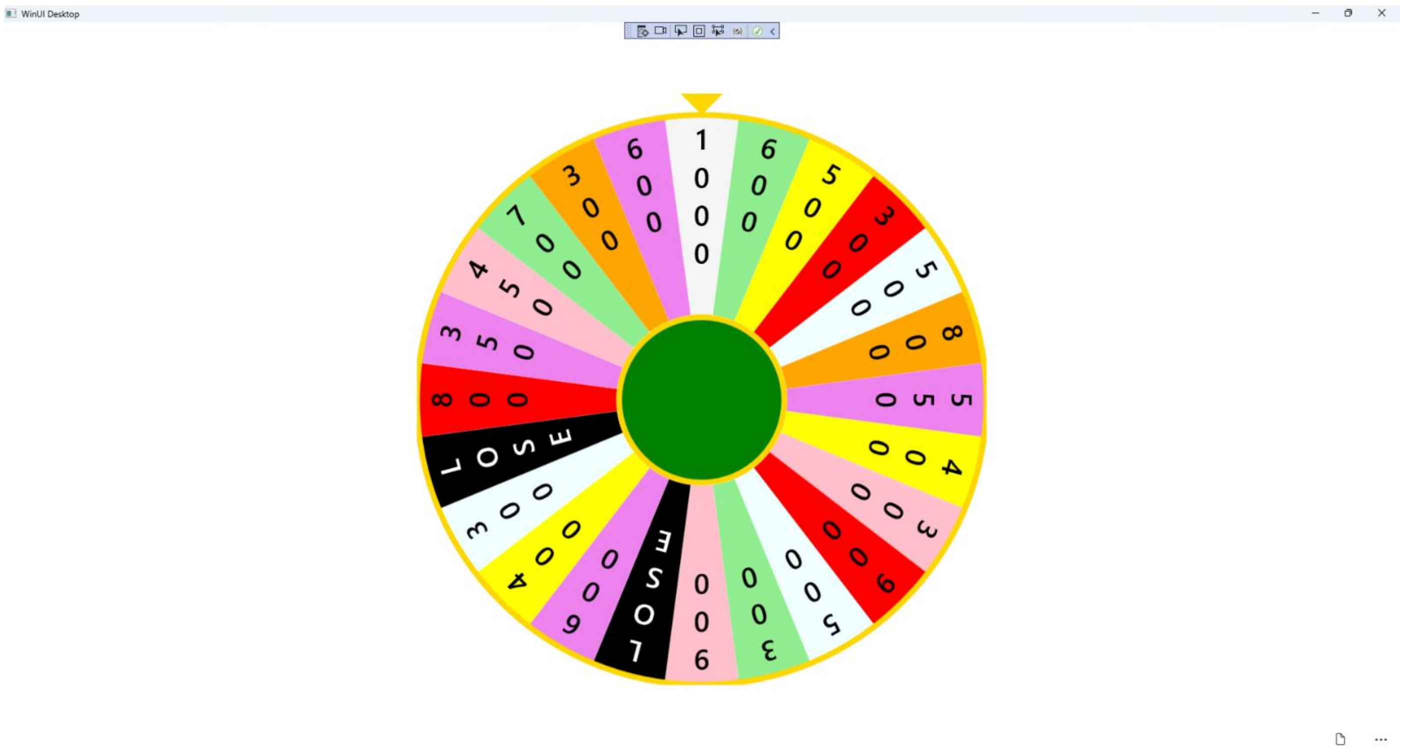
Step 18

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



Step 19

Once running you can then select anywhere on the wheel and it will start spinning and once it stops on a value that's what you win, but if it lands on **LOSE** the game will be over or select *New* to restart the game.



Step 20

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!