tutorialr.com

tutorial

Windows App SDK

Lucky Bingo

tutorialr.com

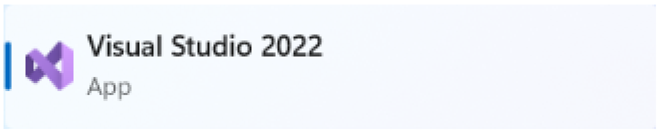
Lucky Bingo

Lucky Bingo shows how you can create a bingo game and display the numbers and balls selected with a control from **NuGet** using the **Windows App SDK**.

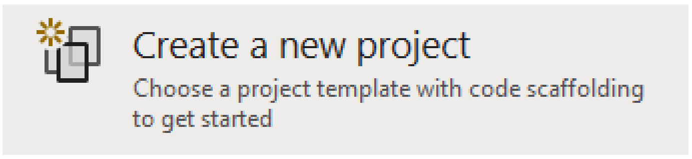
Step 1

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

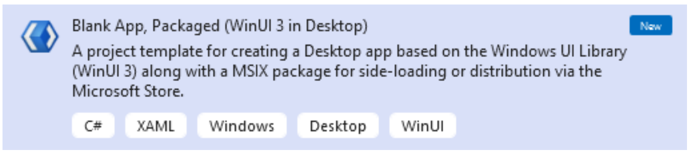
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



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

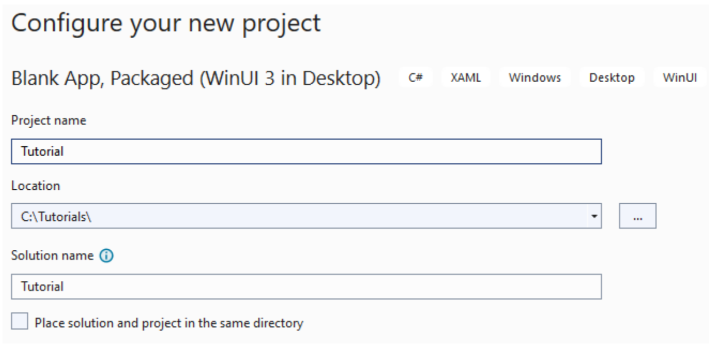

Create a new project
Choose a project template with code scaffolding to get started

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 *LuckyBingo*, 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

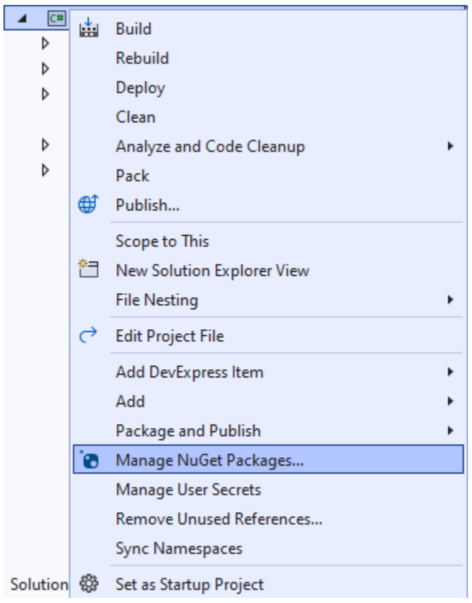
Location: C:\Tutorials\

Solution name: Tutorial

Place solution and project in the same directory

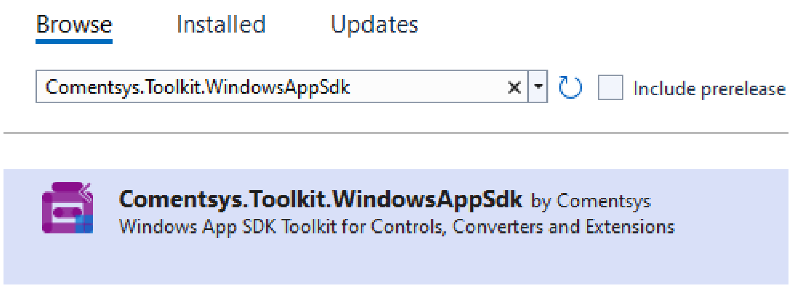
Step 2

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

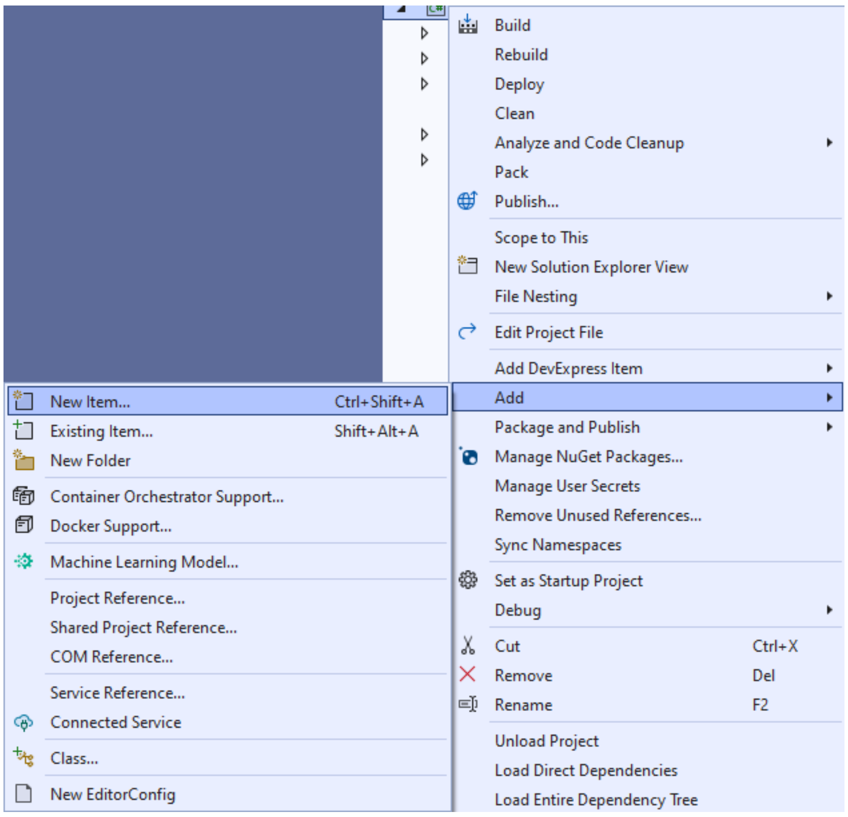
Then in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Toolkit.WindowsAppSdk** and then select **Comentsys.Toolkit.WindowsAppSdk by Comentsys** as indicated and select **Install**


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: LuckyBingo** by selecting the **x** next to it.

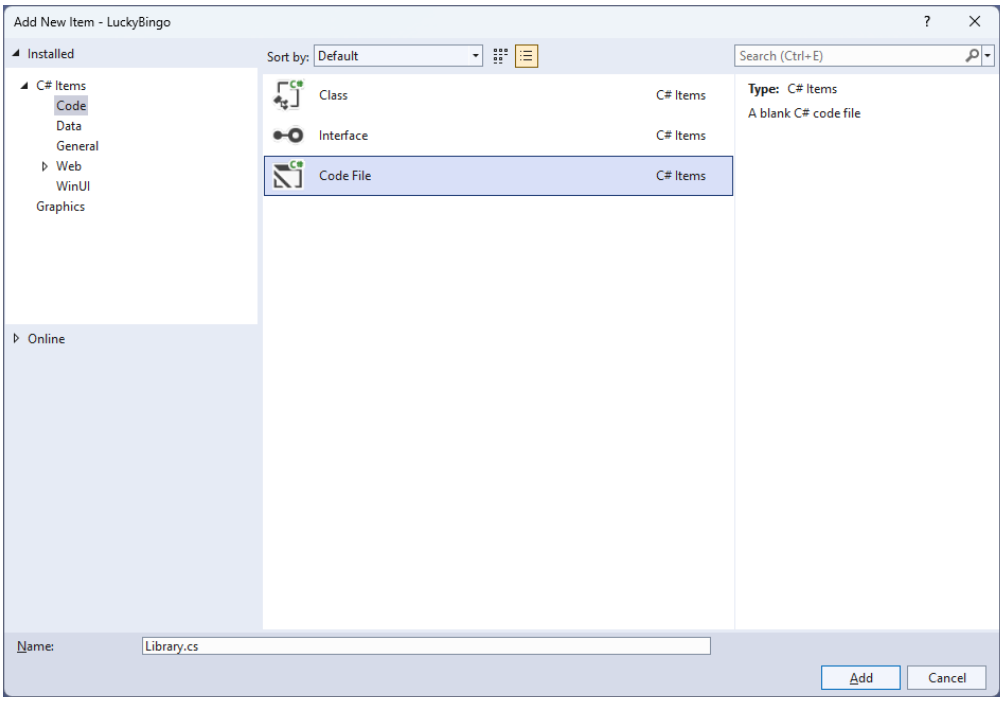
Step 4

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**.



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.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Media;
using System;
using System.Collections.Generic;
using System.Linq;
using Windows.UI;

public class Library
{
    private const string title = "Lucky Bingo";
    private const int size = 22;
    private const int balls = 90;
    private const int marks = 25;
    private static readonly Dictionary<int, Color> _style = new()
    {
        { 0, Colors.DarkViolet },
        { 10, Colors.DeepSkyBlue },
        { 20, Colors.Green },
        { 30, Colors.Gold },
        { 40, Colors.DarkOrange },
        { 50, Colors.RoyalBlue },
        { 60, Colors.Crimson },
        { 70, Colors.DarkCyan },
        { 80, Colors.Purple }
    };
    private readonly Random _random = new((int)DateTime.UtcNow.Ticks);

    private int _count;
    private int _house;
    private List<int> _balls;
    private List<int> _marks;
    private bool _over = false;
    private Dialog _dialog;
    private StackPanel _panel = new();

    // Choose, Piece & Add


    // Layout, Ball & Mark


    // New & Play

}
```

The **`Class`** that has been defined in so far *Library.cs* has **`using`** amongst others for the package of **`Comentsys.Toolkit.WindowsAppSdk`**. It also has colours to use for the bingo balls along with **Variables** to be used with the in the game.

Step 7

While still in the **`Class`** for *Library.cs* and after the **Comment** of **`// Choose, Piece & Add`** type in the following **Methods**:

```
private List<int> Choose(int minimum, int maximum, int total) =>
    Enumerable.Range(minimum, maximum)
        .OrderBy(r => _random.Next(minimum, maximum))
            .Take(total).ToList();

private Piece Piece(bool isBall, int value) => new()
{
    Width = size,
    Height = size,
    Value = $"{value}",
    IsSquare = !isBall,
    Opacity = isBall ? 0 : 1,
    Stroke = new SolidColorBrush(isBall ? _style
        .Where(w => value > w.Key)
        .Select(s => s.Value)
        .LastOrDefault() : Colors.Red),
    Name = isBall ? $"ball{value}" : $"mark{value}"
};

private void Add(ref Grid grid, bool isBall, int row, int column, int value)
{
    var counter = Piece(isBall, value);
    counter.SetValue(Grid.RowProperty, row);
    counter.SetValue(Grid.ColumnProperty, column);
    grid.Children.Add(counter);
}
```

`Choose` will be used to select a unique list of randomised numbers and **`Piece`** will be used to create the bingo ball and numbers on the bingo card. **Add** will be used to position the bingo balls or bingo card numbers using **`Piece`**.

Step 8

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Layout, Ball & Mark`** type in the following **Methods**.

```
private Grid Layout(bool isBall, int rows, int columns, List<int> list)
{
    int count = 0;
    Grid grid = new()
    {
        Margin = new Thickness(5),
        VerticalAlignment = VerticalAlignment.Center
    };
    // Setup Grid
    for (int row = 0; row < rows; row++)
    {
        grid.RowDefinitions.Add(new RowDefinition());
    }
    for (int column = 0; column < columns; column++)
    {
        grid.ColumnDefinitions.Add(new ColumnDefinition());
    }
    // Setup Board
    for (int row = 0; row < rows; row++)
    {
        for (int column = 0; column < columns; column++)
        {
            Add(ref grid, isBall, row, column, list[count]);
            count++;
        }
    }
    return grid;
}

private void Ball(int value)
{
    UIElement element = (UIElement)_panel.FindName($"ball{value}");
    if (element != null) element.Opacity = 1;
}

private void Mark(int value)
{
    Piece piece = (Piece)_panel.FindName($"mark{value}");
    if (piece != null) piece.Fill = piece.Stroke;
}
```

`Layout` will create the layout of either the set of bingo balls to be displayed that will be selected during the game, or it will create the layout of the bingo numbers on the bingo card in the game. **`Ball`** and **`Mark`** will be used to either show the balls for the bingo game selected or mark of the numbers on the bingo card as they're matched.

Step 9

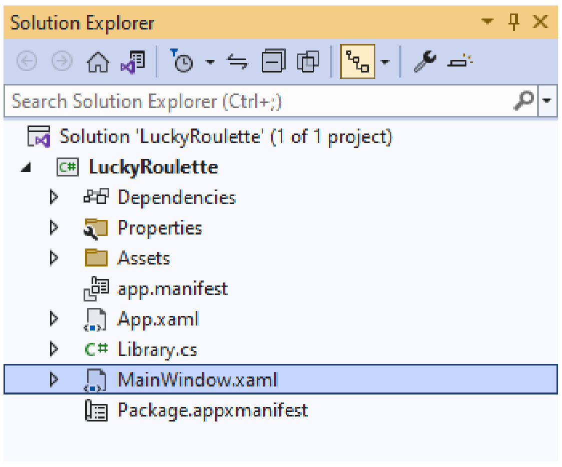
While still in the **`Class`** for *Library.cs* after the **Comment** of **`// New & Play`** type in the following **Methods**.

```
public void New(StackPanel panel)
{
    _count = 0;
    _house = 0;
    _over = false;
    _panel = panel;
    _balls = Choose(1, balls, balls);
    _marks = Choose(1, balls, marks);
    _dialog = new Dialog(panel.XamlRoot, title);
    panel.Children.Clear();
    panel.Children.Add(Layout(true, 9, 10, _balls));
    panel.Children.Add(Layout(false, 5, 5, _marks));
}

public void Play(StackPanel panel)
{
    if (!panel.Children.Any()) New(panel);
    if (_count < balls && !_over)
    {
        var ball = _balls[_count];
        Ball(ball);
        if (_marks.Contains(ball))
        {
            _house++;
            Mark(ball);
            if (_house == marks)
            {
                _over = true;
                _dialog.Show($"Full House in {_count} Balls!");
            }
        }
        _count++;
    }
    else
    {
        _dialog.Show($"Game Over!");
    }
}
```

These **Methods** will be used to start the game with **`New`** which will generate the look-and-feel of the game and **`Play`** will be used to proceed to the next selected bingo ball to allow you to proceed at your own pace until all the numbers on the bingo card are matched.

Step 10

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



Step 11

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 12

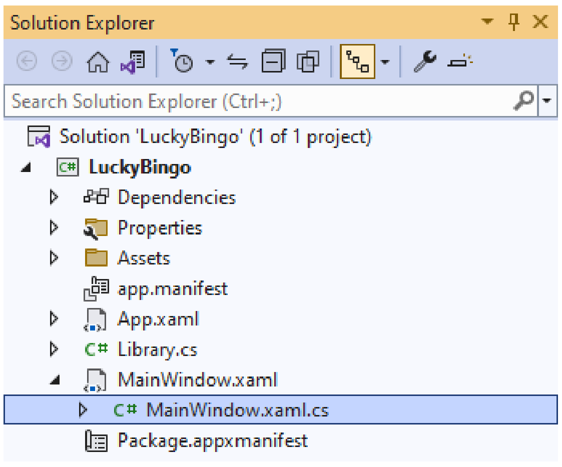
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

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

This **XAML** contains a **`Grid`** with a **`Viewbox`** which will **Scale** a **`StackPanel`**. It has a **Loaded** event handler for **New** which is also shared by the **`AppBarButton`** for *New* and there is the **`AppBarButton`** for *Play* to pick the next bingo ball.

Step 13

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 14

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 15

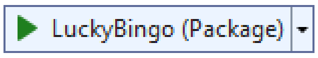
Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

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

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

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

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

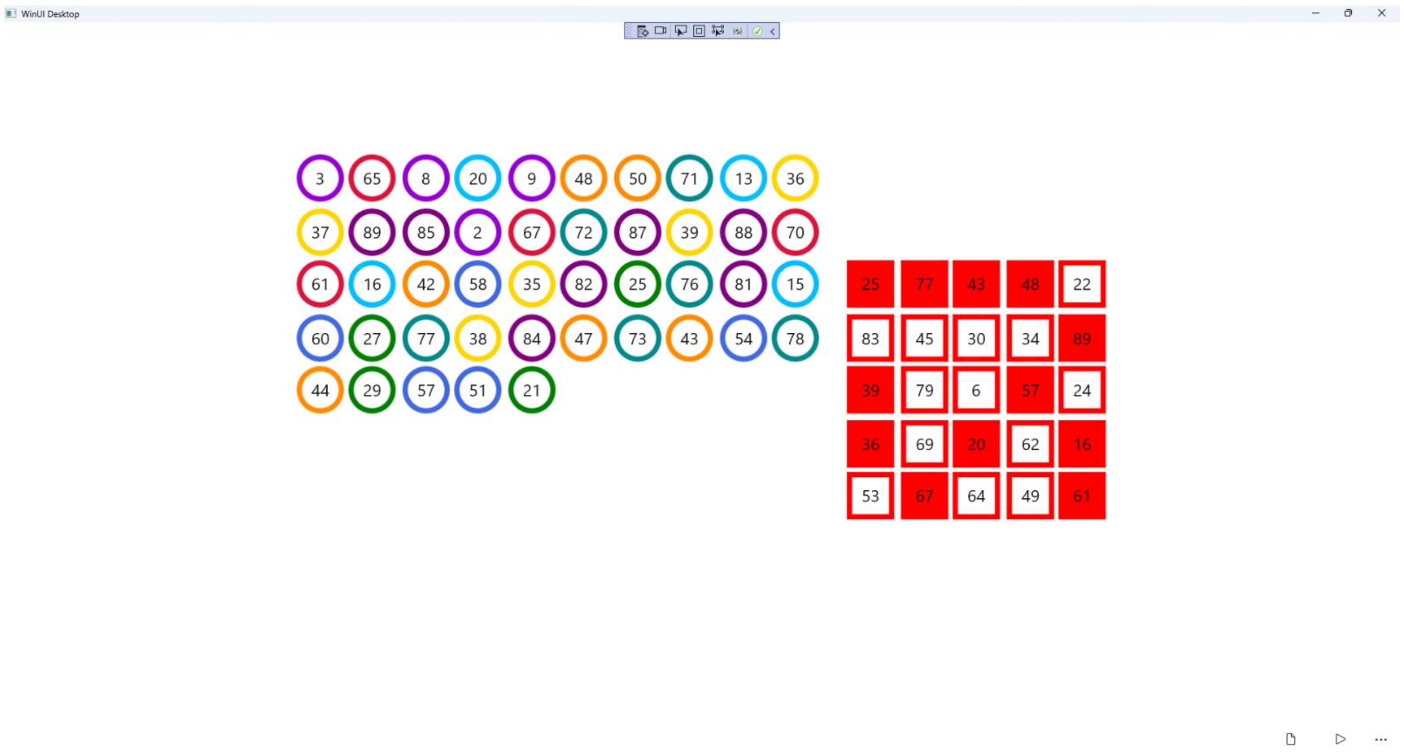
Step 16

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



Step 17

Once running you should see the bingo card then you can click *Play* which will select a random bingo ball, if this matches the bingo card it will be marked off, mark off all the numbers for a full house or you can select *New* to restart the game.



Step 18

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!