

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

Windows App SDK

Lucky Roshambo





# Lucky Roshambo

**Lucky Roshambo** shows how you can create simple **Rock-Paper-Scissors** game or **Roshambo** as it is

known in parts of North America, using emoji and with 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. | 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 *LuckyRoshambo*, 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 **Manage NuGet Packages…**

Graphical user interface, application

Description automatically generated

## 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**

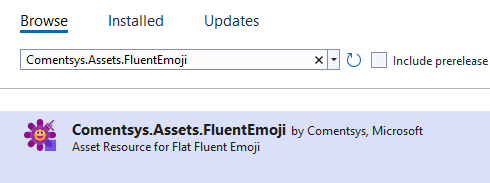
Graphical user interface, text, application, email

Description automatically generated

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.

## Step 4

Then while still in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Assets.FluentEmoji** and then select **Comentsys.Assets.FluentEmoji by Comentsys** as indicated and select **Install**



This will add the package for **Comentsys.Assets.FluentEmoji** 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: LuckyRoshambo** by selecting the **x** next to it.

## Step 5

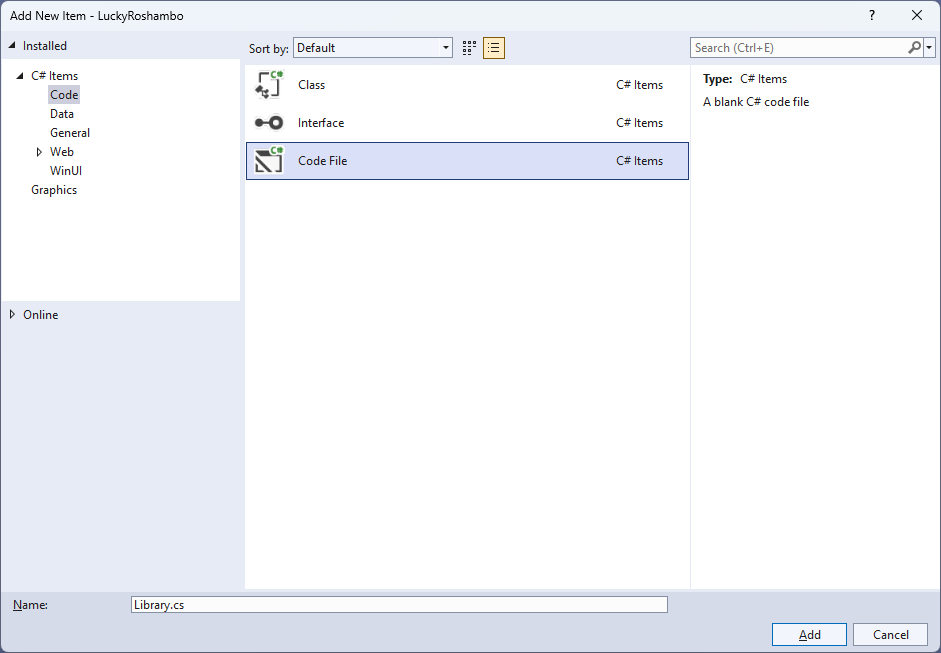
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

Table

Description automatically generated with low confidence

## Step 6

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 7

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

using Comentsys.Assets.FluentEmoji;

using Comentsys.Toolkit.WindowsAppSdk;

using Microsoft.UI.Xaml;

using Microsoft.UI.Xaml.Controls;

using System;

public class Library

{

private const string title = "Lucky Roshambo";

private const int size = 3;

private const int lost = 0;

private const int win = 1;

private const int draw = 2;

private static readonly int[,] \_match = new int[size, size]

{

{ draw, lost, win },

{ win, draw, lost },

{ lost, win, draw }

};

private static readonly FluentEmojiType[] \_assets = new FluentEmojiType[]

{

FluentEmojiType.Rock,

FluentEmojiType.PageWithCurl,

FluentEmojiType.Scissors

};

private static readonly string[] \_values = new string[]

{

"You Lost!",

"You Win!",

"You Draw!"

};

private readonly Random \_random = new((int)DateTime.UtcNow.Ticks);

private Dialog \_dialog;

// Asset & Play

// Get & New

}

The **Class** that has been defined in so far *Library.cs* has **using** for the packages that were addedof **Comentsys.Assets.FluentEmoji** and **Comentsys.Toolkit.WindowsAppSdk** amongst others needed. There are also some **const** and **readonly** values for parts of the game such as the **FluentEmojiType** for *Rock*, *Page with Curl* for *Paper and Scissors* and to represent the board along with a **Dialog** that will be used to display messages in the game.

## Step 8

Still in the **Class** for *Library.cs* after the **Comment** of **// Asset & Play** type the following **Methods**:

private Viewbox Asset(int asset) => new()

{

Width = 100,

Height = 100,

Child = new Asset

{

AssetResource = FlatFluentEmoji.Get(\_assets[asset])

}

};

private void Play(int option)

{

int computer = \_random.Next(0, size - 1);

var result = \_match[option, computer];

var content = new StackPanel()

{

Orientation = Orientation.Vertical

};

content.Children.Add(new TextBlock()

{

HorizontalTextAlignment = TextAlignment.Center,

Text = "Computer Picked"

});

content.Children.Add(Asset(computer));

content.Children.Add(new TextBlock()

{

HorizontalTextAlignment = TextAlignment.Center,

Text = \_values[result]

});

\_dialog.Show(content);

}

**Asset** will be used to get the relevant asset to display the *Rock*, *Paper* or *Scissors*and **Play** will be used when it is time to see if can beat the selection of *Rock*, *Paper* or *Scissors* with your own selection and will display a message showing what the result is.

## Step 9

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

private Button Get(int option)

{

Button button = new()

{

Width = 150,

Height = 150,

Tag = option,

Content = Asset(option),

Margin = new Thickness(5)

};

button.Click += (object sender, RoutedEventArgs e) =>

Play((int)((Button)sender).Tag);

return button;

}

public void New(StackPanel panel)

{

\_dialog = new Dialog(panel.XamlRoot, title);

panel.Children.Clear();

for (int index = 0; index < size; index++)

{

panel.Children.Add(Get(index));

}

}

**Get** is used to obtain a **Button** and set the **Event** for **Click** to use the **Method** for **Play** and **New** is used to create the look-and-feel for the game and to start the game.

## 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"/>

</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**.

## 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);

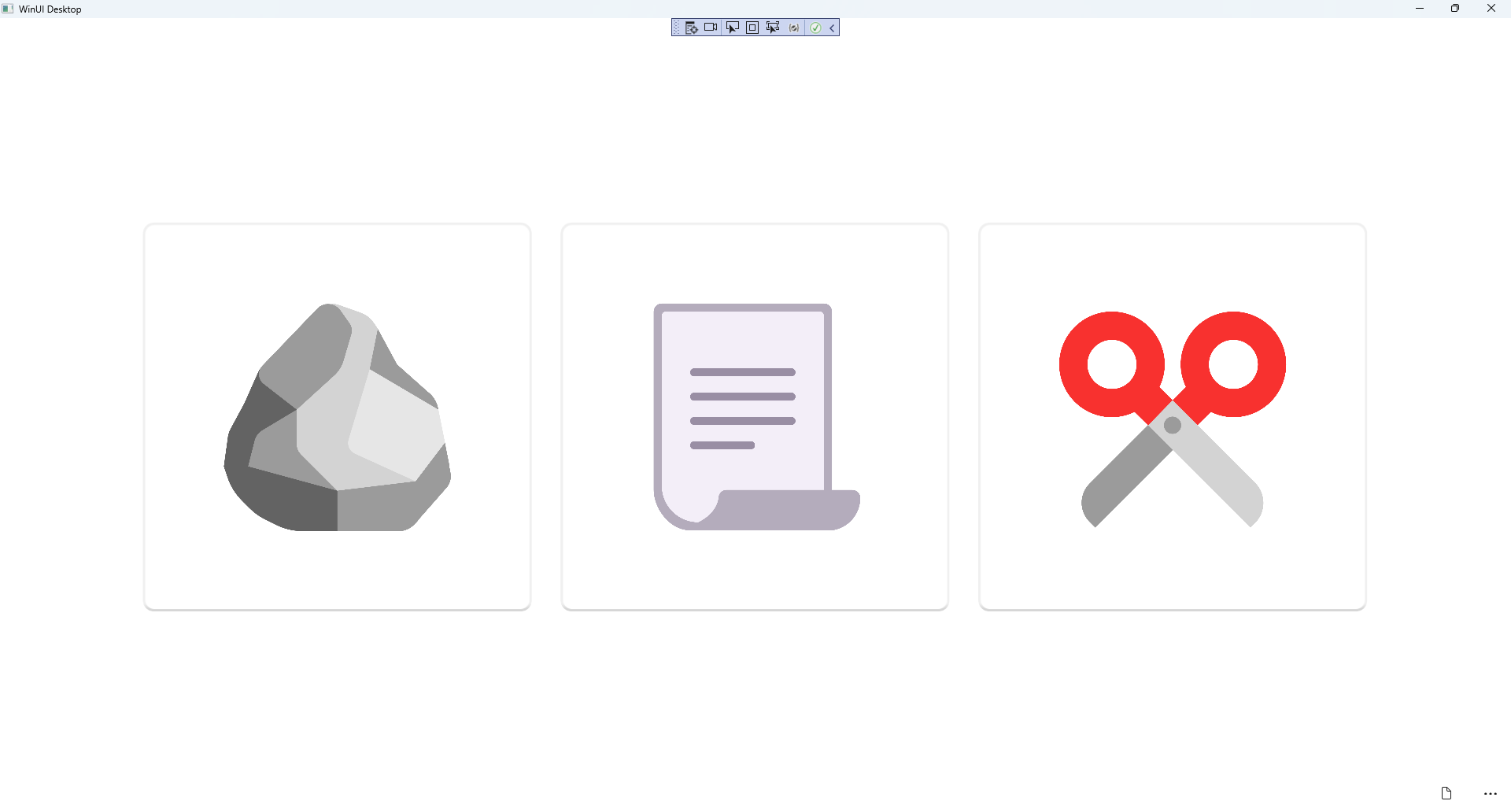
Here an **Instance** of the **Class** of **Library** is created then below this is the **Method** of **New** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **=>** for an Expression Bodywhich is useful when a **Method** only has one line.

## Step 16

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

## Step 17

Once running you can then press on the first button - **Rock**, the second button for **Paper** or the third button for **Scissors** then you can see what the **Computer** selects to see if you **Win**, **Lose** or **Draw** or you can restart the game by selecting *New*.

****

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