

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

Windows App SDK

Hit or Miss





# Hit or Miss

**Hit or Miss** shows how you can create a simple random game where you can either score a **Hit** or a **Miss**

displayed with 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 *HitOrMiss*, 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**

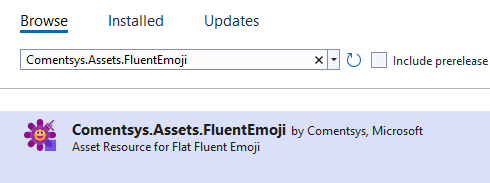
Graphical user interface, text, application, email

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

## Step 5

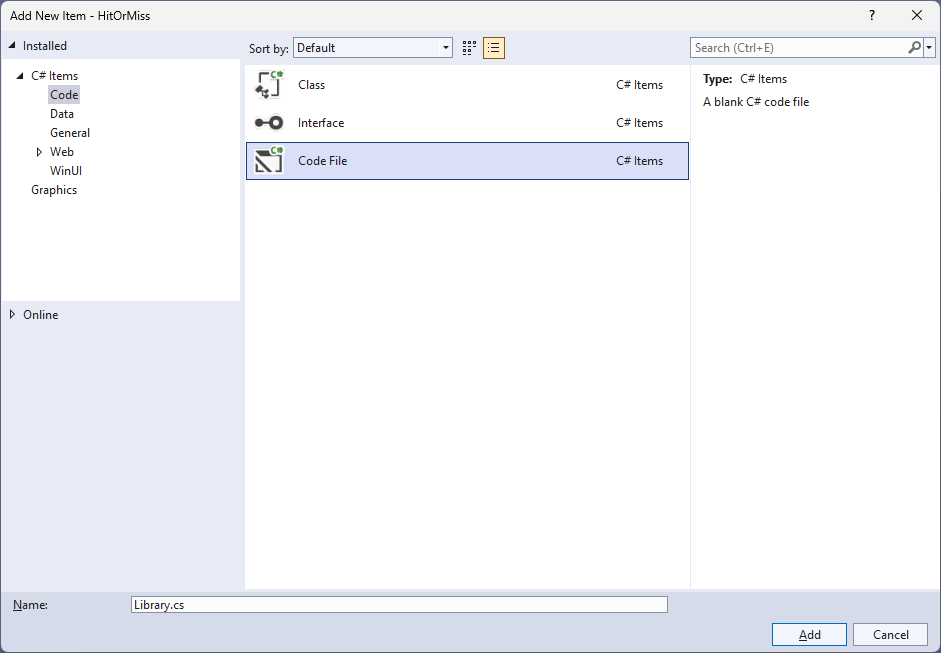
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

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;

using System.Collections.Generic;

using System.Linq;

public class Library

{

private const string title = "Hit or Miss";

private const int score = 18;

private const int size = 6;

private const string hit = "X";

private const string miss = "O";

private readonly string[,] \_board = new string[size, size];

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

private int \_go = 0;

private int \_hits = 0;

private int \_misses = 0;

private bool \_won = false;

private Dialog \_dialog;

private List<int> Choose(int minimum, int maximum, int total) =>

Enumerable.Range(minimum, maximum)

.OrderBy(r => \_random.Next(minimum, maximum))

.Take(total).ToList();

private Viewbox Asset(string value) => new()

{

Child = new Asset()

{

AssetResource = FlatFluentEmoji.Get(

value == hit ? FluentEmojiType.Collision :

FluentEmojiType.Hole)

}

};

// Add

// Layout & 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 and to represent the board then there are **Methods** of **Choose** which is used to select a random list of numbers and **Asset** which will represent **Emoji** that will be used for the hits and misses.

## Step 8

Still in the **Class** for *Library.cs* after the **Comment** of **// Add** type the following **Method**:

private void Add(ref Grid grid, int row, int column)

{

Button button = new()

{

Width = 64,

Height = 64

};

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

{

if (!\_won)

{

button = (Button)sender;

string selected = \_board[

(int)button.GetValue(Grid.RowProperty),

(int)button.GetValue(Grid.ColumnProperty)

];

if (button.Content == null)

{

button.Content = Asset(selected);

if (selected == hit)

\_hits++;

else if (selected == miss)

\_misses++;

\_go++;

}

if (\_go < (size \* size) && \_misses < score)

{

if (\_hits == score)

{

\_dialog.Show(

$"You Won! With {\_hits} hits and {\_misses} misses");

\_won = true;

}

}

else

{

\_dialog.Show($"You Lost! With {\_hits} hits and {\_misses} misses");

\_won = true;

}

}

};

button.SetValue(Grid.ColumnProperty, column);

button.SetValue(Grid.RowProperty, row);

grid.Children.Add(button);

}

**Add** is used to add a **Button** which when clicked, set with the **Event Handler** of **Click** will indicate if the selection was a hit or a miss with a message to indicate whether the game was won or lost.

## Step 9

While still in the **Class** for *Library.cs* after the **Comment** of **// Layout & New** type in the following **Methods** of **Layout** which will layout the board for the game and **New** which will start a new game.

private void Layout(Grid grid)

{

\_go = 0;

\_hits = 0;

\_misses = 0;

grid.Children.Clear();

grid.RowDefinitions.Clear();

grid.ColumnDefinitions.Clear();

// Setup Grid

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

{

grid.RowDefinitions.Add(new RowDefinition());

grid.ColumnDefinitions.Add(new ColumnDefinition());

}

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

{

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

{

Add(ref grid, row, column);

}

}

}

public void New(Grid grid)

{

Layout(grid);

\_won = false;

int index = 0;

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

// Setup Values

List<string> values = new();

while (values.Count < (size \* size))

{

values.Add(hit);

values.Add(miss);

}

List<int> indices = Choose(1, size \* size, size \* size);

// Setup Board

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

{

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

{

\_board[column, row] = values[indices[index] - 1];

index++;

}

}

}

## 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>

<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 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);

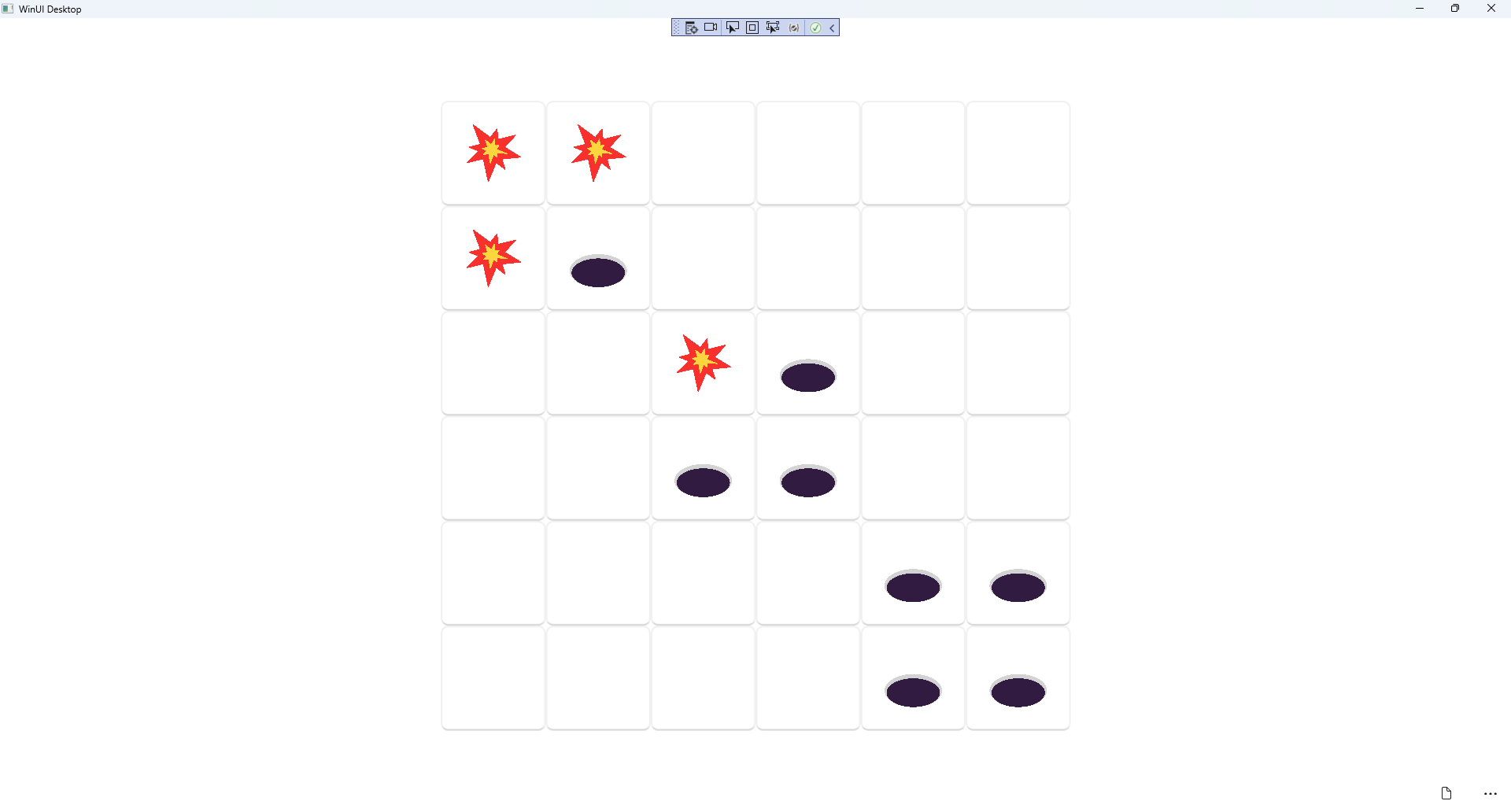
Here an **Instance** of the **Class** of **Library** is created then below this is the **Method** of **New** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **=>** for an Expression Bodywhich is useful when a **Method** only has one line.

## Step 16

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

## Step 17

Once running you can start by clicking on any **Button**, to win you will need to get more hits (**Collisions**) than misses (**Holes**) up to a total of *18* to win! 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)! |  |