

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

Windows App SDK

Touch Game





# Touch Game

**Touch Game** shows how you can create a pattern matching game using 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 *TouchGame*, 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, then you can close the **tab** for **Nuget: TouchGame** by selecting the **x** next to it.

## Step 4

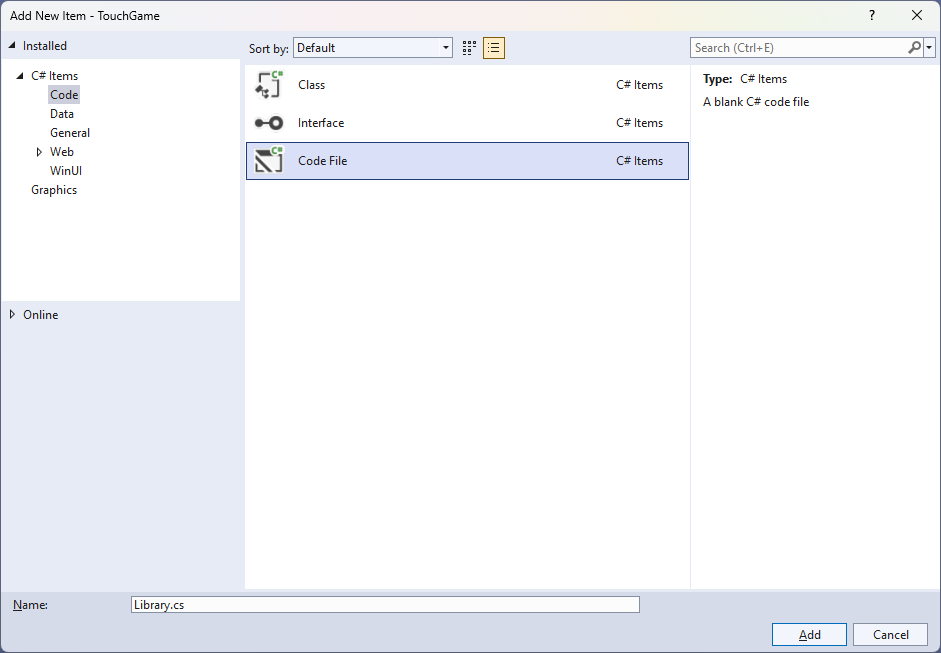
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

Table

Description automatically generated with low confidence

## 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**.



## 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 System.Threading.Tasks;

using Windows.UI;

public class Library

{

private const string title = "Touch Game";

private const int size = 2;

private const int level = 10;

private const int delay\_duration = 250;

private const int timer\_duration = 500;

private static readonly Dictionary<int, Color> \_options = new()

{

{ 0, Colors.Red },

{ 1, Colors.Blue },

{ 2, Colors.Green },

{ 3, Colors.Gold }

};

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

private Grid \_grid;

private int \_score;

private bool \_over;

private int \_index;

private bool \_playing = false;

private Dialog \_dialog;

private DispatcherTimer \_timer;

private List<int> \_values = new();

// Choose, Option & Set

// Play

// Add & Layout

// Tick & New

}

**Class** defined so far *Library.cs* has **using** for packageof **Comentsys.Toolkit.WindowsAppSdk** and others. It also has **Constants** to represent things needed in the game and there are **Variables** to keep track of values used in the game and elements for the look-and-feel of the game.

## Step 7

Still in the **Class** for *Library.cs* after the **Comment** of **// Choose, Option & Set** type the following **Methods**:

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

{

var choose = new List<int>();

var values = Enumerable.Range(minimum, maximum).ToList();

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

{

var value = \_random.Next(0, values.Count);

choose.Add(values[value]);

}

return choose;

}

private Viewbox Option(int option) => new()

{

Child = new Piece()

{

IsSquare = true,

Name = $"{option}",

Stroke = new SolidColorBrush(\_options[option])

}

};

private async void Set(int option)

{

var piece = \_grid.FindName($"{option}") as Piece;

piece.Fill = piece.Stroke;

await Task.Delay(delay\_duration);

piece.Fill = null;

}

**Choose** is used to select a set of random numbers that are not unique. **Option** is used to get the **Piece** needed to show the pattern and **Set** is used to update the **Piece** to indicate a part of the pattern.

## Step 8

While still in the **Class** for *Library.cs* after the **Comment** of **// Play** type in the following **Method**:

private void Play(int option)

{

if (!\_playing)

{

if (!\_over)

{

var correct = \_values[\_index] == option;

if (correct)

{

if (\_index < \_score)

\_index++;

else

{

\_score++;

if (\_score < level)

{

\_index = 0;

\_timer.Start();

}

else

\_over = true;

}

}

else

\_over = true;

}

if (\_over)

\_dialog.Show($"Game Over! You scored {\_score} out of {level}!");

}

}

**Play** is used set the option for the pattern along with checking to see how many correct patterns have been performed and will display a **Dialog** the score if the pattern is not correct or the game is over.

## Step 9

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

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

{

Button button = new()

{

Width = 100,

Height = 100,

Tag = option,

Content = Option(option),

Margin = new Thickness(5)

};

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

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

button.SetValue(Grid.ColumnProperty, column);

button.SetValue(Grid.RowProperty, row);

grid.Children.Add(button);

}

private void Layout(Grid grid)

{

grid.Children.Clear();

grid.RowDefinitions.Clear();

grid.ColumnDefinitions.Clear();

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

{

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

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

}

int count = 0;

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

{

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

{

Add(grid, row, column, count);

count++;

}

}

}

**Add** is used to setup the **Buttons** to input the pattern and will use the **Method** for **Play** when the event handler for **Click** is triggered and **Layout** will use **Add** to create the look-and-feel for the game.

## Step 10

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

private void Tick()

{

if (\_index <= \_score)

{

\_playing = true;

Set(\_values[\_index]);

\_index++;

}

else

{

\_index = 0;

\_timer.Stop();

\_playing = false;

}

}

public void New(Grid grid)

{

\_index = 0;

\_score = 0;

\_grid = grid;

\_over = false;

Layout(grid);

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

\_values = Choose(0, 3, level);

\_timer = new DispatcherTimer()

{

Interval = TimeSpan.FromMilliseconds(timer\_duration)

};

\_timer.Tick += (object sender, object e) =>

Tick();

\_timer.Start();

}

**Tick** will display the pattern that needs to be duplicated using **Set** and **New** will setup and start a game.

## Step 11

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

## Step 12

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 13

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 14

|  |  |
| --- | --- |
| 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 15

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 16

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);

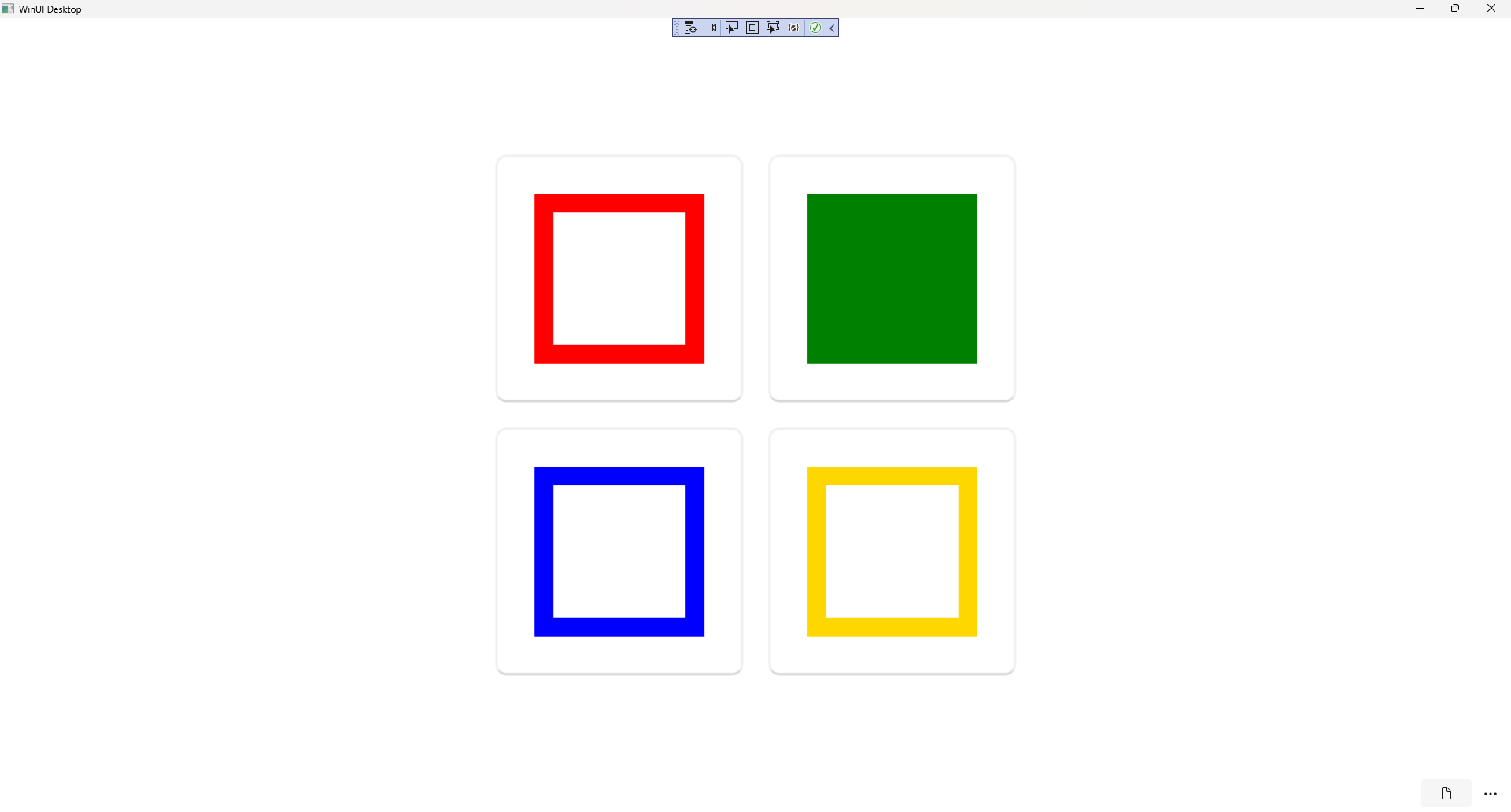
Here an **Instance** of the **Class** of **Library** is created then below this is the **Method** of **New** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **=>** for an Expression Bodywhich is useful when a **Method** only has one line.

## Step 17

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

## Step 18

Once running you can then select any **Button** then one of the **Squares** will be highlighted, select the correct one, then each time one more **Square** will be highlighted each turn, match the pattern to continue but if you get it wrong you lose, or you can select *New*to start a new game.

****

## Step 19

|  |  |
| --- | --- |
| 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)! |  |