

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

Windows App SDK

Light Game





# Light Game

**Light Game** how to create a simple game to toggle all the squares to **Yellow** from **Black** 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 *LightGame*, 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: LightGame** by selecting the **x** next to it.

## Step 4

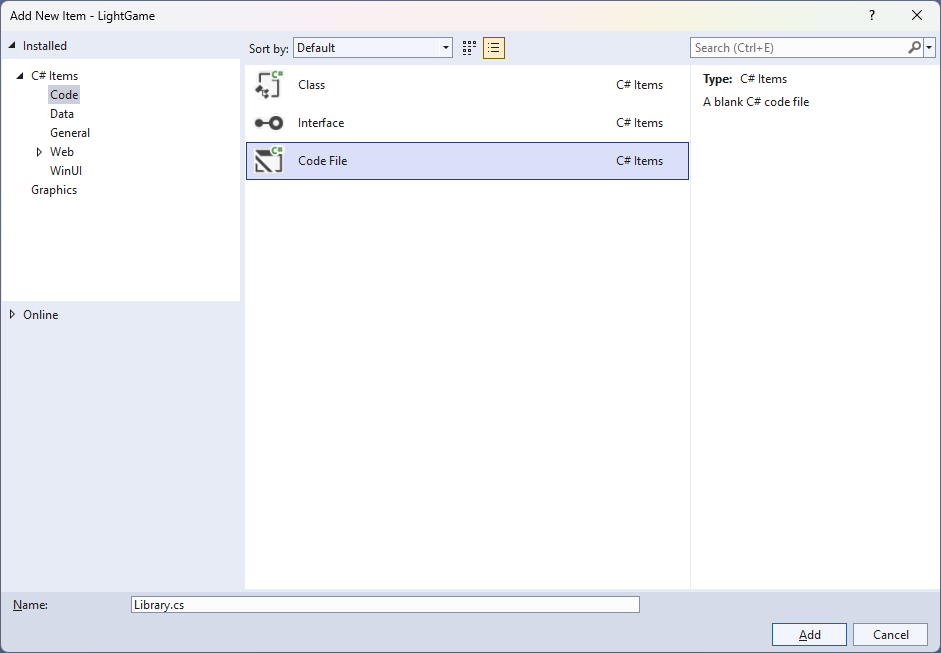
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

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.Controls;

using Microsoft.UI.Xaml.Input;

using Microsoft.UI.Xaml.Media;

using Windows.UI;

public class Library

{

private const string title = "Light Game";

private const int on = 1;

private const int off = 0;

private const int size = 7;

private readonly Color lightOn = Colors.Gold;

private readonly Color lightOff = Colors.Black;

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

private Grid \_grid;

private Dialog \_dialog;

private int \_moves = 0;

private bool \_over = false;

// Toggle, Set & Winner

// Select & Add

// Layout & 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 **// Toggle, Set & Winner** type the following **Methods**:

private void Toggle(int row, int column)

{

\_board[row, column] = \_board[row, column] == on ? off : on;

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

piece.Fill = \_board[row, column] == on ?

new SolidColorBrush(lightOn) :

new SolidColorBrush(lightOff);

}

private void Set(int row, int column)

{

Toggle(row, column);

if (row > 0)

Toggle(row - 1, column); // Toggle Left

if (row < (size - 1))

Toggle(row + 1, column); // Toggle Right

if (column > 0)

Toggle(row, column - 1); // Toggle Above

if (column < (size - 1))

Toggle(row, column + 1); // Toggle Below

}

private bool Winner()

{

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

{

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

{

if (\_board[column, row] == on)

{

return false;

}

}

}

return true;

}

**Toggle** is used to set the elements of the game between being **On** or **Yellow** or **Off** or **Black** then **Set** is used to toggle the elements of the game to the **Left**, **Right**, **Top** and **Bottom** of the element that has been toggled and **Winner** checks to see if the game is over.

## Step 8

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

private void Select(Piece piece)

{

if (!\_over)

{

int row = (int)piece.GetValue(Grid.RowProperty);

int column = (int)piece.GetValue(Grid.ColumnProperty);

Set(row, column);

\_moves++;

if (Winner())

{

\_dialog.Show($"Well Done! You won in {\_moves} moves!");

\_over = true;

}

}

else

\_dialog.Show($"Game Over!");

}

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

{

Piece piece = new()

{

Width = 50,

Height = 50,

IsSquare= true,

Name = $"{row}:{column}",

Fill = new SolidColorBrush(lightOn)

};

piece.Tapped += (object sender, TappedRoutedEventArgs e) =>

Select(piece);

piece.SetValue(Grid.ColumnProperty, column);

piece.SetValue(Grid.RowProperty, row);

grid.Children.Add(piece);

}

**Select** is used to check if the game is over and if not then will use **Set** to toggle elements of the game, then uses **Winner** to check if the game has been won.

## Step 9

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

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());

}

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

{

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

{

Add(grid, row, column);

}

}

}

public void New(Grid grid)

{

\_grid = grid;

\_moves = 0;

\_over = false;

Layout(grid);

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

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

{

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

{

\_board[column, row] = on;

}

}

}

**Layout** will create the look-and-feel for the game and **New** will start a 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>

<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);

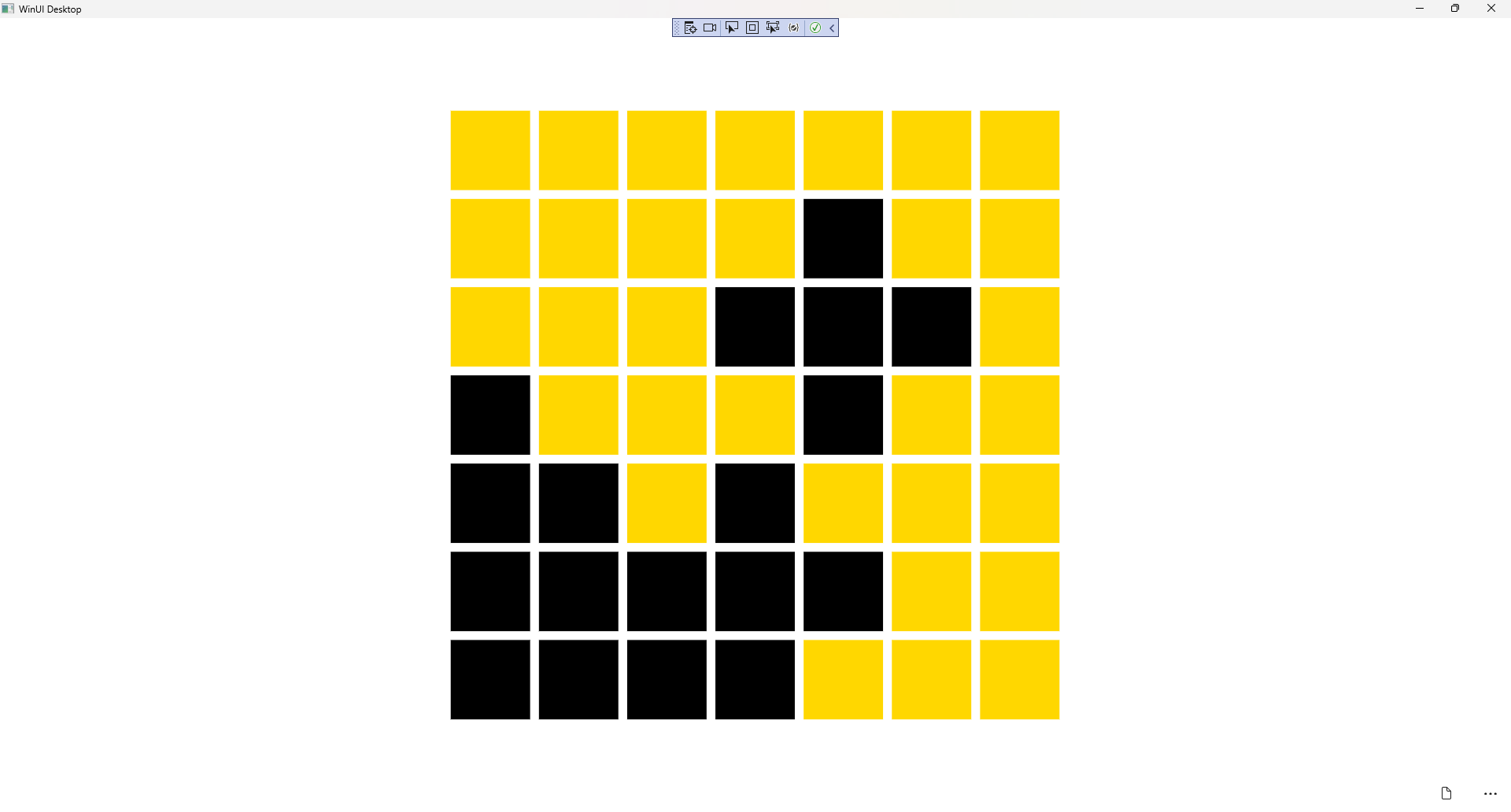
Here an **Instance** of the **Class** of **Library** is created then below this is the **Method** of **New** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **=>** for an Expression Bodywhich is useful when a **Method** only has one line.

## Step 16

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

## Step 17

Once running you can then select any **Square** and you win by setting all the **Squares** that are **Yellow** to **Black** or you can select *New*to start a new 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](https://tutorialr.com)! |  |