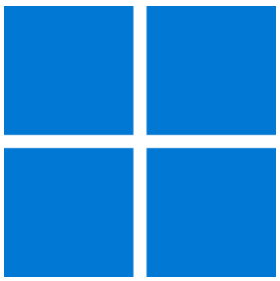
tutorial

Windows App SDK



Four in Row

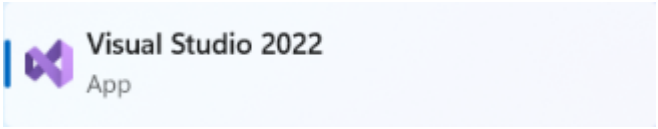
Four in Row

Four in Row shows how you can create simple a two-player game where the objective is to get four items in a horizontal, vertical or diagonal row displayed with emoji and with a toolkit from **NuGet** using the **Windows App SDK**.

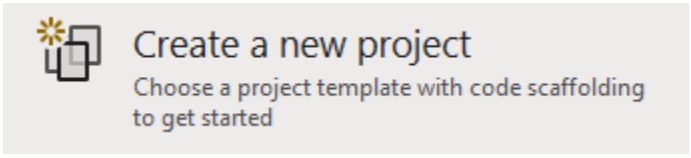
Step 1

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

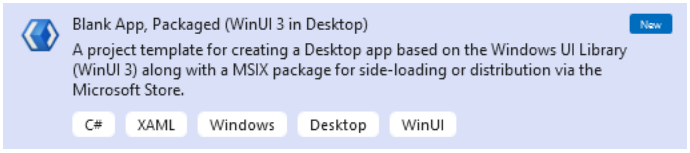
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



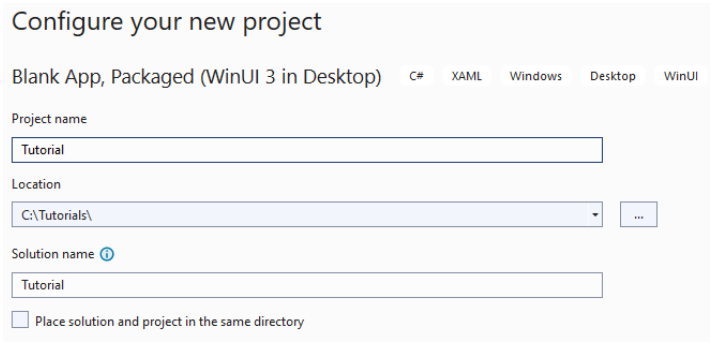
Once **Visual Studio 2022** has started select **Create a new project**.



Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.

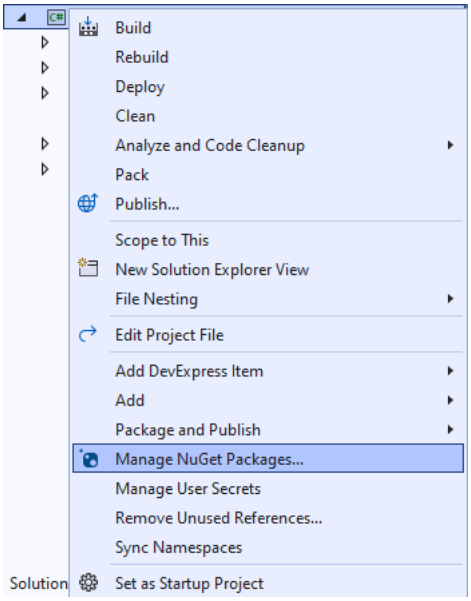


After that in **Configure your new project** type in the **Project name** as *FourInRow*, then select a Location and then select **Create** to start a new **Solution**.



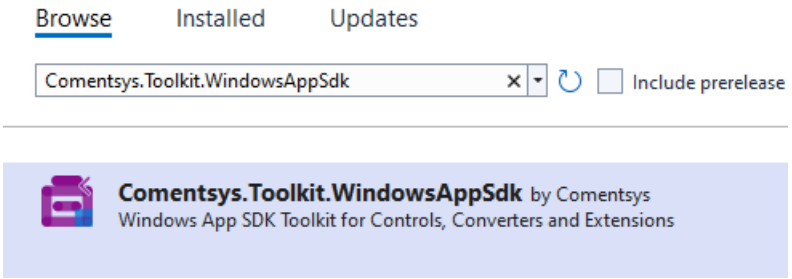
Step 2

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

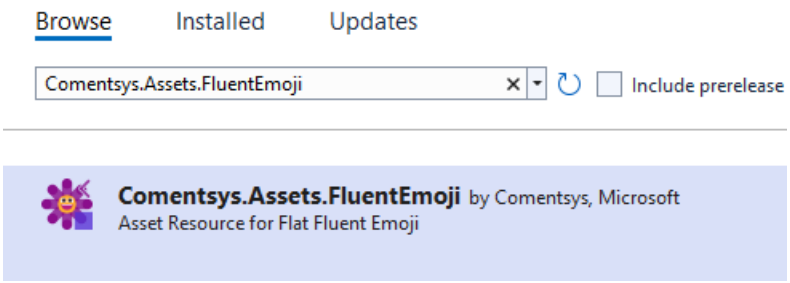
Then in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Toolkit.WindowsAppSdk** and then select **Comentsys.Toolkit.WindowsAppSdk by Comentsys** as indicated and select **Install**



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

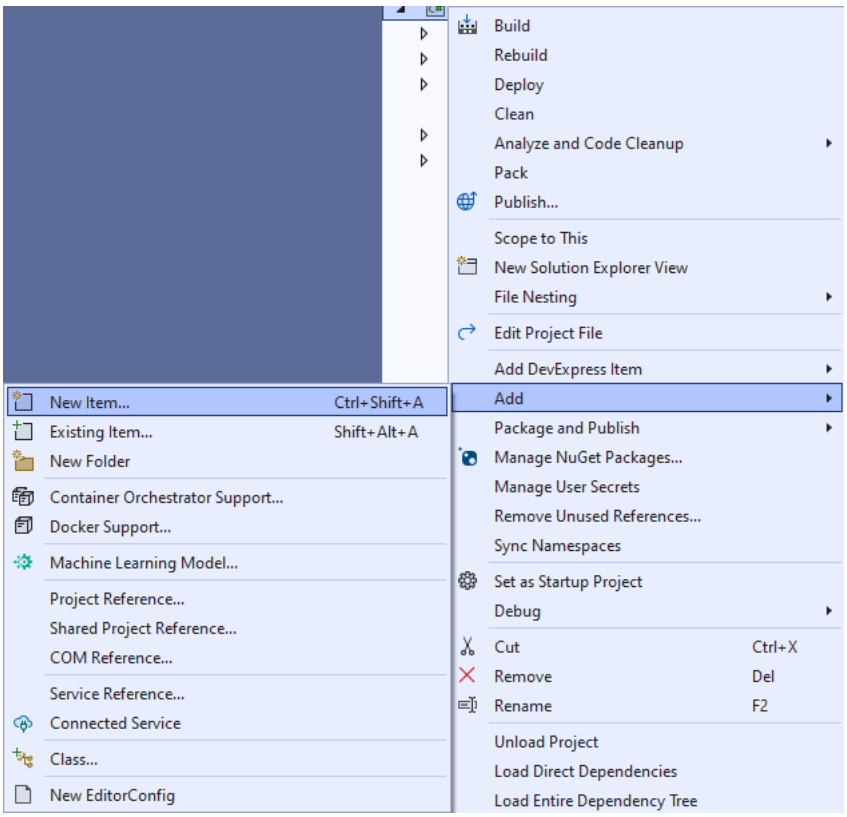
Then while still in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Assets.FluentEmoji** and then select **Comentsys.Assets.FluentEmoji by Comentsys** as indicated and select **Install**



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

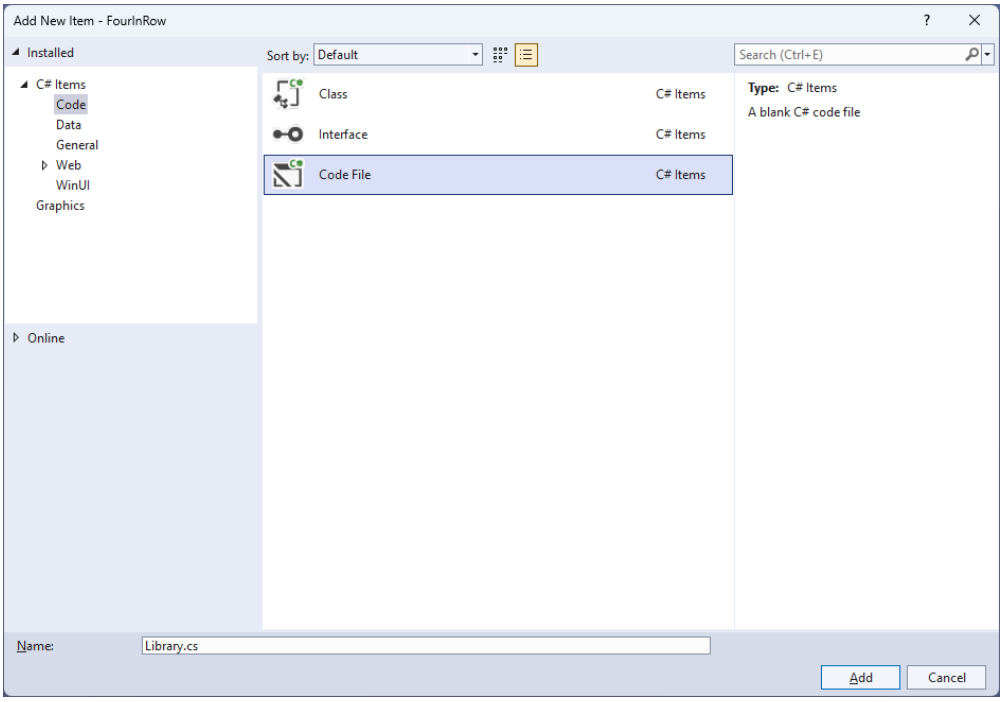
Step 5

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...**



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.Linq;

public class Library
{
    private const string title = "Four In Row";
    private const int total = 3;
    private const int size = 7;
    private readonly string[] _players = { string.Empty, "Yellow", "Red" };
    private readonly int[,] _board = new int[size, size];

    private int _value = 0;
    private int _amend = 0;
    private int _player = 0;
    private bool _won = false;
    private Dialog _dialog;

    // Check Vertical & Check Horizontal

    // Check Diagonal Top Left & Check Diagonal Top Right

    // Winner, Full & Asset

    // Set & Add

    // Layout & New

}
```

The **`Class`** that has been defined in so far *Library.cs* has **`using`** for the packages that were added of **`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 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 **`// Check Vertical & Check Horizontal`** type the following **Methods**:

```
private bool CheckVertical(int row, int column)
{
    _value = 0;
    do
    {
        _value++;
    }
    while (row + _value < size &&
    _board[column, row + _value] == _player);
    return _value > total;
}

private bool CheckHorizontal(int row, int column)
{
    _value = 0;
    _amend = 0;
    // From Left
    do
    {
        _value++;
    }
    while (column - _value >= 0 &&
    _board[column - _value, row] == _player);
    if (_value > total)
        return true;
    // Deduct Middle - Prevent double count
    _value -= 1;
    // Then Right
    do
    {
        _value++;
        _amend++;
    }
    while (column + _amend < size &&
    _board[column + _amend, row] == _player);
    return _value > total;
}
```

`CheckVertical` will check to see if there is a vertical set of four items for the current player or for a horizontal set of four items then will use **`CheckHorizontal`**.

Step 9

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Check Diagonal Top Left & Check Diagonal Top Right`** type in the following **Methods** to check for a set of four diagonal items for a player:

```
private bool CheckDiagonalTopLeft(int row, int column)
{
    _value = 0;
    _amend = 0;
    // From Top Left
    do
    {
        _value++;
    }
    while (column - _value >= 0 && row - _value >= 0 &&
    _board[column - _value, row - _value] == _player);
    if (_value > total)
        return true;
    _value -= 1; // Deduct Middle - Prevent double count
    // To Bottom Right
    do
    {
        _value++;
        _amend++;
    }
    while (column + _amend < size && row + _amend < size &&
    _board[column + _amend, row + _amend] == _player);
    return _value > total;
}

private bool CheckDiagonalTopRight(int row, int column)
{
    _value = 0;
    _amend = 0;
    // From Top Right
    do
    {
        _value++;
    }
    while (column + _value < size && row - _value >= 0 &&
    _board[column + _value, row - _value] == _player);
    if (_value > total)
        return true;
    _value -= 1; // Deduct Middle - Prevent double count
    // To Bottom Left
    do
    {
        _value++;
        _amend++;
    }
    while (column - _amend >= 0 &&
    row + _amend < size &&
    _board[column - _amend,
    row + _amend] == _player);
    return _value > total;
}
```

Step 10

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Winner, Full & Asset`** type in the following **Methods**:

```
private bool Winner(int row, int column)
{
    bool vertical = CheckVertical(row, column);
    bool horizontal = CheckHorizontal(row, column);
    bool diagonalTopLeft = CheckDiagonalTopLeft(row, column);
    bool diagonalTopRight = CheckDiagonalTopRight(row, column);
    return vertical || horizontal ||
    diagonalTopLeft || diagonalTopRight;
}

private bool Full()
{
    for (int row = 0; row < size; row++)
    {
        for (int column = 0; column < size; column++)
        {
            if (_board[column, row] == 0)
            {
                return false;
            }
        }
    }
    return true;
}

private Viewbox Asset(int player) => new()
{
    Child = new Asset()
    {
        AssetResource = FlatFluentEmoji.Get(
        player == 1 ? FluentEmojiType.YellowCircle :
        FluentEmojiType.RedCircle)
    }
};
```

`Winner` will use the previous **Methods** to check if the current player is the winner to see if there is a vertical, horizontal or diagonal set of items. **`Full`** will be used to check if the board is full and **`Asset`** will be used to create the **Emoji** to represent the players with a *Yellow Circle* and a *Red Circle*.

Step 11

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Set & Add`** type in the following **Methods** which are **`Set`** to place an item for a player and **`Add`** for the **`Button`** and **Events** which form the game board.

```
private void Set(Grid grid, int row, int column)
{
    for (int i = size - 1; i > -1; i--)
    {
        if (_board[column, i] == 0)
        {
            _board[column, i] = _player;
            Button button = (Button)grid.Children.Single(
            w => Grid.GetRow((Button)w) == i
            && Grid.GetColumn((Button)w) == column);
            button.Content = Asset(_player);
            row = i;
            break;
        }
    }
    if (Winner(row, column))
    {
        _won = true;
        _dialog.Show($"{_players[_player]} has won!");
    }
    else if (Full())
        _dialog.Show("Board Full!");
    _player = _player == 1 ? 2 : 1; // Set Player
}

private void Add(Grid grid, int row, int column)
{
    Button button = new()
    {
        Width = 100,
        Height = 100,
        Name = $"{row}:{column}"
    };
    button.Click += (object sender, RoutedEventArgs e) =>
    {
        if (!_won)
        {
            button = (Button)sender;
            row = (int)button.GetValue(Grid.RowProperty);
            column = (int)button.GetValue(Grid.ColumnProperty);
            if (_board[column, 0] == 0) // Check Free Row
                Set(grid, row, column);
        }
        else
            _dialog.Show("Game Over!");
    };
    button.SetValue(Grid.ColumnProperty, column);
    button.SetValue(Grid.RowProperty, row);
    grid.Children.Add(button);
}
```

Step 12

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

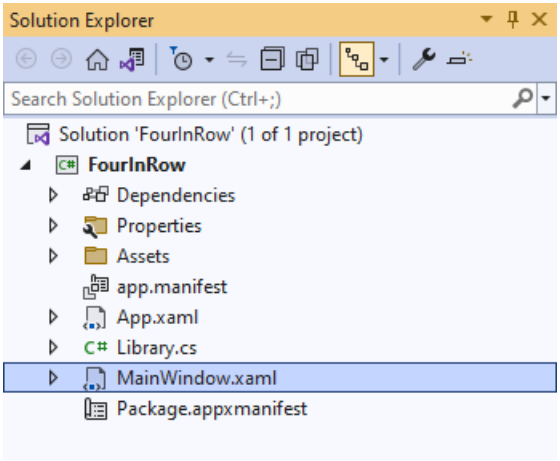
```
private void Layout(Grid grid)
{
    grid.Children.Clear();
    grid.ColumnDefinitions.Clear();
    grid.RowDefinitions.Clear();
    // Setup Grid
    for (int index = 0; index < size; index++)
    {
        grid.RowDefinitions.Add(new RowDefinition());
        grid.ColumnDefinitions.Add(new ColumnDefinition());
    }
    // Setup Board
    for (int column = 0; column < size; column++)
    {
        for (int row = 0; row < size; row++)
        {
            Add(grid, row, column);
            _board[row, column] = 0;
        }
    }
}

public async void New(Grid grid)
{
    _won = false;
    _dialog = new Dialog(grid.XamlRoot, title);
    _player = await _dialog.ConfirmAsync("Who goes First?",
        _players[1], _players[2]) ? 1 : 2;
    Layout(grid);
}
```

Layout creates the look-and-feel of the game by setting out the game board and **New** will start a new game and ask which player should go first as either *Yellow* or *Red*.

Step 13

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



Step 14

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 15

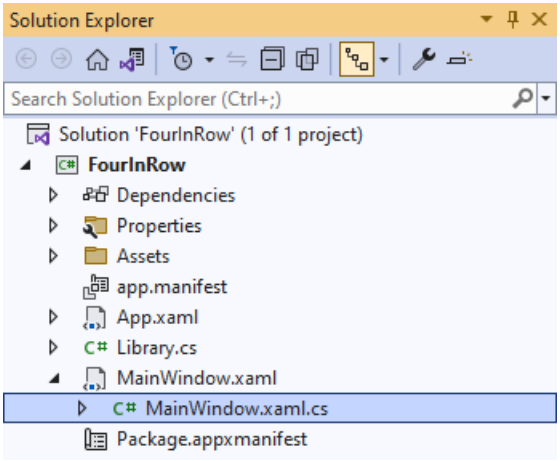
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<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 16

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 17

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 18

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

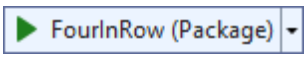
```
private readonly Library _library = new();

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 Body which is useful when a **Method** only has one line.

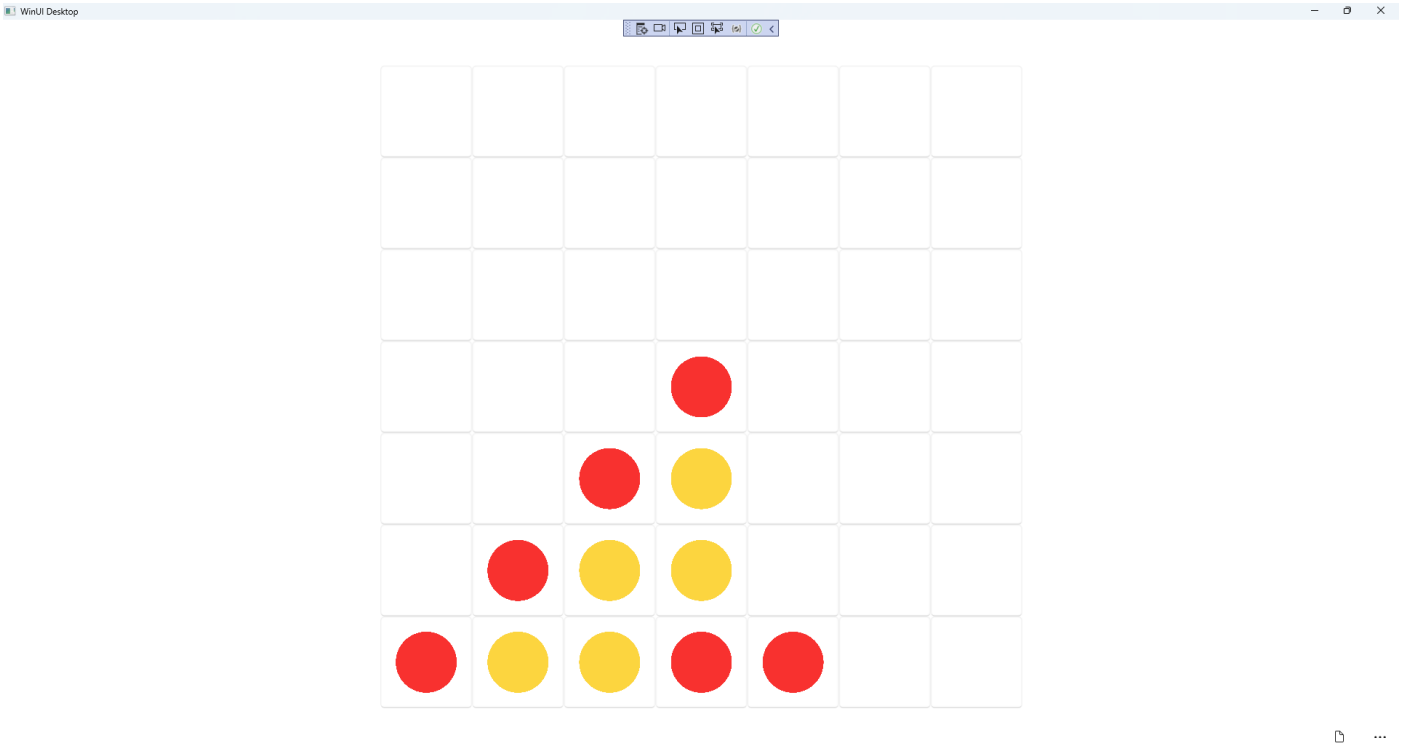
Step 19

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



Step 20

Once running you can choose to play as *Yellow* or *Red* then the first player to get a horizontal, vertical or diagonal set of items wins the game or you can restart the game by selecting *New*.



Step 21

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!