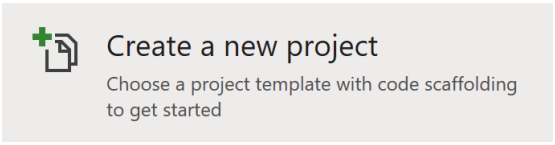
Universal Windows Platform – Four in Row

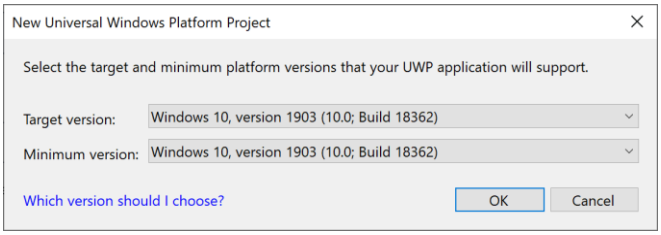
Four in Row shows how to create a simple two-player game where the objective is to get four pieces in a row in either horizontal, vertical or diagonal directions

Step 1



Follow **Setup and Start** on how to Install and/or Get Started with **Visual Studio 2019** if not already or in **Windows 10** choose **Start**, find and select **Visual Studio 2019** then from the **Get started** screen select **Create a new project**

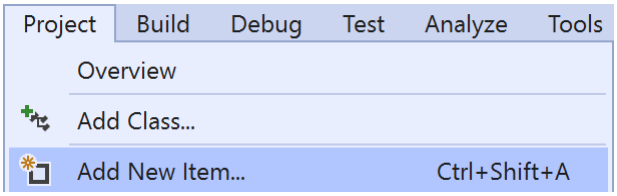
Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **FourInRow** and select **Create**



Finally, in **New Universal Windows Platform Project** pick the **Target version** and **Minimum version** to be at least **Windows 10, version 1903 (10.0; Build 18362)** and then select **OK**

Target Version will control the most recent features of Windows 10 your application can use. To make sure you always have the most recent version, check for any Notifications or Updates in Visual Studio 2019

Step 2



Choose **Project** then **Add New Item...** from the **Menu** in **Visual Studio 2019**

Step 3

Then choose **Code File** from **Add New Item** in **Visual Studio 2019**, enter the **Name** as **Library.cs** and select **Add**

Universal Windows Platform – Four in Row

Step 4

In the **Code** View of **Library.cs** will be displayed and in this the following should be entered:

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;

public class Library
{
    private const string title = "Four In Row";
    private const string yellow = "\U0001F7E1";
    private const string red = "\U0001F534";
    private const int total = 3;
    private const int size = 7;
    private readonly string[] _players =
    {
        string.Empty, "Yellow", "Red"
    };

    private int _value = 0;
    private int _amend = 0;
    private int _player = 0;
    private bool _won = false;
    private int[,] _board = new int[size, size];


}
```

There are `using` statements to include necessary functionality. Also there are `private const` for the setup of the game and for the values that will represent the look-and-feel of the game, there are also `private` members to store values for the game including the `_players` and the `_board`

Then below the **`private int[,] _board = new int[size, size];`** line the following **methods** should be entered:

```
private void Show(string content, string title)
{
    _ = new MessageDialog(content, title).ShowAsync();
}

private async Task<bool> ConfirmAsync(string content, string title,
    string ok, string cancel)
{
    bool result = false;
    MessageDialog dialog = new MessageDialog(content, title);
    dialog.Commands.Add(new UICommand(ok,
        new UICommandInvokedHandler((cmd) => result = true)));
    dialog.Commands.Add(new UICommand(cancel,
        new UICommandInvokedHandler((cmd) => result = false)));
    await dialog.ShowAsync(); return result;
}
```

`Show` method is used to display a basic `MessageDialog` and `ConfirmAsync` is used to display a `MessageDialog` with an `ok` and `cancel` option

Next below the **`private async Task<bool> ConfirmAsync(...) { ... }` method** the following **method** should be entered:

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

`CheckVertical` method is used check the `_board` has a set of four vertical `_player` items in the `_board`

Universal Windows Platform – Four in Row

Next after the **`private bool CheckVertical(...) { ... }` method** the following **method** should be entered:

```
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);
    if (_value > total)
    {
        return true;
    }
    return false;
}
```

`CheckHorizontal` method is used check the `_board` has a set of four horizontal `_player` items in the `_board`

Universal Windows Platform – Four in Row

Then after the **`private bool CheckHorizontal(...) { ... }` method** the following **method** should be entered:

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

`CheckDiagonalTopLeft` method is used check the `_board` has a set of four diagonal `_player` items in the `_board` from top left to bottom right

Universal Windows Platform – Four in Row

Next after the **`private bool CheckDiagonalTopLeft(...) { ... }` method** the following **method** should be entered:

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

`CheckDiagonalTopRight` method is used check the `_board` has a set of four diagonal `_player` items in the `_board` from top right to bottom left

Then after the **`private bool CheckDiagonalTopRight(...) { ... }` method** the following **methods** should be entered:

```
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;
}
```

`Winner` method will use all the check methods to see if there is a winner in either `vertical`, `horizontal`, `diagonalTopLeft` or `diagonalTopRight` directions. `Full` will check if the `_board` is full

Next after **the `private bool Full() { ... }` method** the following **method** should be entered:

```
private Viewbox Piece(int player)
{
    TextBlock textblock = new TextBlock()
    {
        IsColorFontEnabled = true,
        Text = player == 1 ? yellow : red,
        TextLineBounds = TextLineBounds.Tight,
        FontFamily = new FontFamily("Segoe UI Emoji"),
        HorizontalTextAlignment = TextAlignment.Center
    };
    return new Viewbox()
    {
        Child = textblock
    };
}
```

`Piece` method is used to create a `TextBlock` for the `player` in the game

Then after the **`private Viewbox Piece(int player)` method** the following **method** should be entered:

```
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 = Piece(_player);
            row = i;
            break;
        }
    }
    if (Winner(row, column))
    {
        _won = true;
        Show($"{_players[_player]} has won!", title);
    }
    else if (Full())
        Show("Board Full!", title);
    _player = _player == 1 ? 2 : 1; // Set Player
}
```

`Set` method is used to call `Piece` to set the `_player` and will check `Winner` method or `Full` method to see if the game has been won, or is over and will change the `_player`

Universal Windows Platform – Four in Row

After the **`private void Set(...)` method** the following **method** should be entered:

```
private void Add(Grid grid, int row, int column)
{
    Button button = new Button()
    {
        Width = 100,
        Height = 100,
        Name = $"{row}:{column}",
        Margin = new Thickness(5),
        Style = (Style)Application.Current.Resources
            ["ButtonRevealStyle"]
    };
    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
            Show("Game Over!", title);
    };
    button.SetValue(Grid.ColumnProperty, column);
    button.SetValue(Grid.RowProperty, row);
    grid.Children.Add(button);
}
```

`Add` method is used to create the elements that will make up the game and will also check `if` the game is over and will call `Set` method to play the game

Next after the **`private void Add(...) { ... }` method** the following **method** should be entered:

```
private void Layout(ref 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;
        }
    }
}
```

`Layout` method is used to create the look-and-feel of the game including setting up the `Grid` by calling the `Add` method

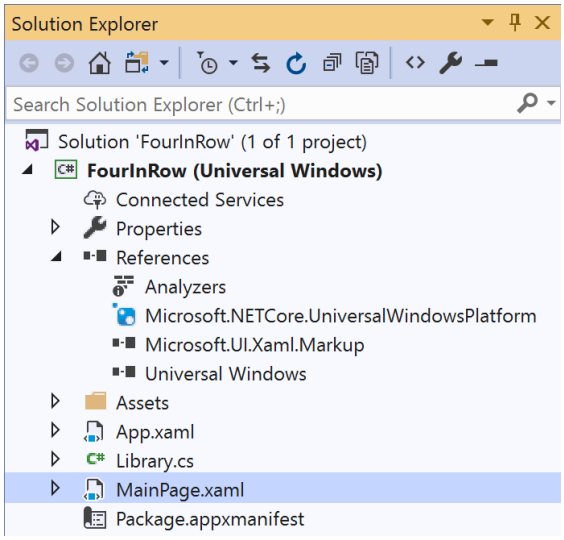
Finally after the **`private void Layout(...) { ... }` method** the following `public` **method** should be entered:

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

`New` method will setup the layout of the `Grid` using the `Layout` method and will use `ConfirmAsync` to choose `Who goes First?`

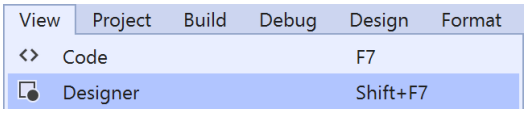
Universal Windows Platform – Four in Row

Step 5



In the **Solution Explorer** of **Visual Studio 2019** select **MainPage.xaml**

Step 6



Choose **View** then **Designer** from the **Menu** in **Visual Studio 2019**

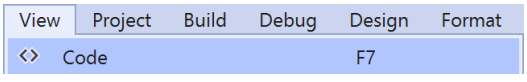
Step 7

In the **Design** View and **XAML** View of **Visual Studio 2019** will be displayed, and in this between the **Grid** and **/Grid** elements enter the following **XAML**:

```
<Viewbox>
    <Grid Margin="50" Name="Display"
    HorizontalAlignment="Center"
    VerticalAlignment="Center"/>
</Viewbox>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Icon="Page2" Label="New" Click="New_Click"/>
</CommandBar>
```

The first block of XAML the main user interface features a Viewbox to contain a Grid which will display the game. The second block of XAML is the CommandBar which contains New to start the game

Step 8



Choose **View** then **Code** from the **Menu** in **Visual Studio 2019**

Universal Windows Platform – Four in Row

Step 9

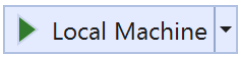
Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
Library library = new Library();

private void New_Click(object sender, RoutedEventArgs e)
{
    library.New(Display);
}
```

Below the MainPage method an instance of the `Library` Class is created. The `New_Click` event handler will call the `New` method in the `Library` class

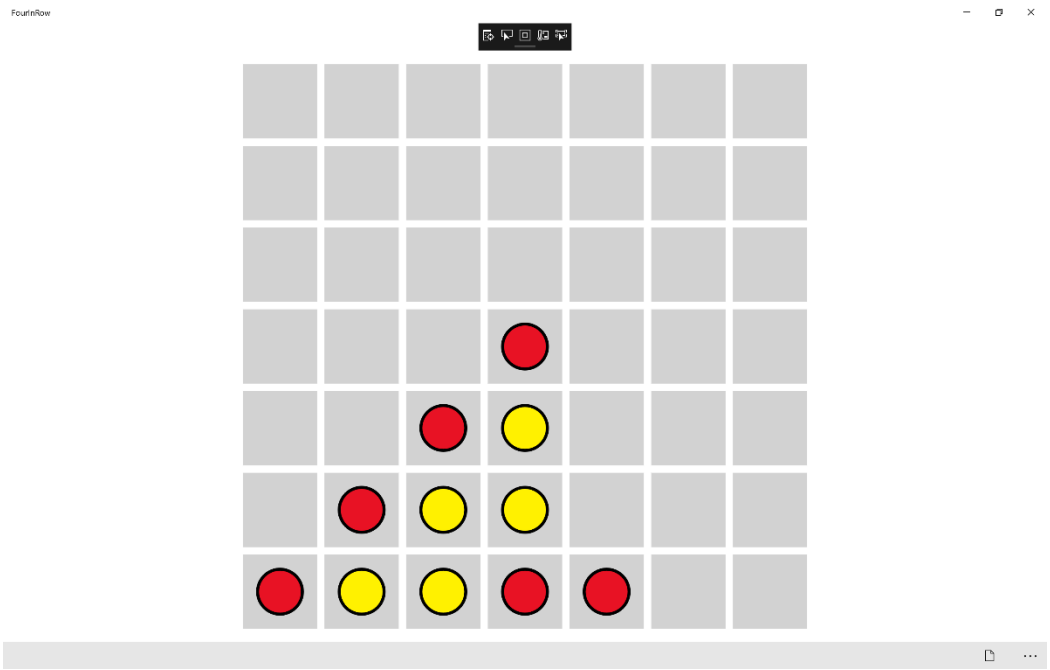
Step 10



That completes the **Universal Windows Platform** Application, in **Visual Studio 2019** select **Local Machine** to run the Application

Step 11

Once the Application is running use **New** to start the playing, first can chose to play as **Red** or **Yellow** and you can win by getting four pieces in a horizontal, vertical or diagonal row



Step 12

To Exit the Application, select the **Close** button in the top right of the Application