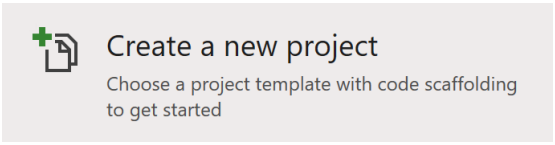
Universal Windows Platform – Tic Tac Toe

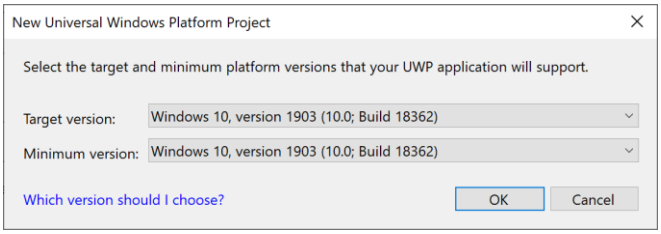
Tic Tac Toe shows how to use a **Grid** and **Emoji** to implement the Game of **Tic Tac Toe** also known as **Noughts and Crosses**

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**

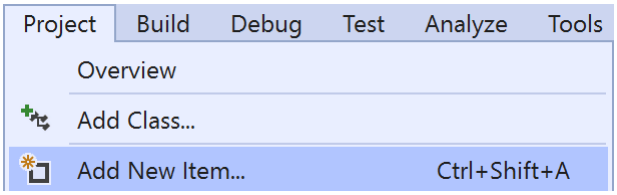
Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **TicTacToe** and select **Create**



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 – Tic Tac Toe

Step 4

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

```
using System;
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 = "Tic Tac Toe";
    private const string blank = " ";
    private const string nought = "\U00002B55";
    private const string cross = "\U0000274C";
    private const int size = 3;

    private bool _won = false;
    private string _piece = blank;
    private string[,] _board = new string[size, size];


}
```

There are `using` statements to include necessary functionality. There are `private const string` for the setup of the game and for the values that will represent the `nought` with and `cross` that makes up the game of Tic Tac Toe. There are `private` members including the two-dimensional array `_board` which represents what will appear in the game

Then below the **`private string[,] _board = new string[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()` is used to display a basic `MessageDialog` and `ConfirmAsync(...)` is used to display a `MessageDialog` with `UICommand` for `ok` and `cancel`

Universal Windows Platform – Tic Tac Toe

Next below **`private async Task<bool> ConfirmAsync { ... }`** enter the following **methods**:

```
private bool Winner()
{
    return (_board[0, 0] == _piece && _board[0, 1] ==
        _piece && _board[0, 2] == _piece) ||
    (_board[1, 0] == _piece && _board[1, 1] ==
        _piece && _board[1, 2] == _piece) ||
    (_board[2, 0] == _piece && _board[2, 1] ==
        _piece && _board[2, 2] == _piece) ||
    (_board[0, 0] == _piece && _board[1, 0] ==
        _piece && _board[2, 0] == _piece) ||
    (_board[0, 1] == _piece && _board[1, 1] ==
        _piece && _board[2, 1] == _piece) ||
    (_board[0, 2] == _piece && _board[1, 2] ==
        _piece && _board[2, 2] == _piece) ||
    (_board[0, 0] == _piece && _board[1, 1] ==
        _piece && _board[2, 2] == _piece) ||
    (_board[0, 2] == _piece && _board[1, 1] ==
        _piece && _board[2, 0] == _piece);
}

private bool Drawn()
{
    return _board[0, 0] != blank && _board[0, 1] !=
        blank && _board[0, 2] != blank &&
    _board[1, 0] != blank && _board[1, 1] !=
        blank && _board[1, 2] != blank &&
    _board[2, 0] != blank && _board[2, 1] !=
        blank && _board[2, 2] != blank;
}
```

`Winner()` is used to determine if a player has won and `Drawn()` is used to determine if the game is a draw

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

```
private Viewbox Piece()
{
    TextBlock textblock = new TextBlock()
    {
        Text = _piece,
        IsColorFontEnabled = true,
        TextLineBounds = TextLineBounds.Tight,
        FontFamily = new FontFamily("Segoe UI Emoji"),
        HorizontalTextAlignment = TextAlignment.Center
    };
    return new Viewbox()
    {
        Child = textblock
    };
}
```

`Piece()` is used to `return` a `Viewbox` which contains a `TextBlock` which will contain the Emoji value of the current player of either `nought` or `cross`

Universal Windows Platform – Tic Tac Toe

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

```
private void Add(ref Grid grid, int row, int column)
{
    Button button = new Button()
    {
        Width = 75,
        Height = 75,
        Margin = new Thickness(10),
        Style = (Style)Application.Current.Resources
        ["ButtonRevealStyle"]
    };
    button.Click += (object sender, RoutedEventArgs e) =>
    {
        if (!_won)
        {
            button = (Button)sender;
            if (button.Content == null)
            {
                button.Content = Piece();
                _board[(int)button.GetValue(Grid.RowProperty),
                (int)button.GetValue(Grid.ColumnProperty)] = _piece;
            }
            if (Winner())
            {
                _won = true;
                Show($"{_piece} wins!", title);
            }
            else if (Drawn())
            {
                Show("Draw!", title);
            }
            else
            {
                // Swap Players
                _piece = (_piece == cross ? nought : cross);
            }
        }
        else
        {
            Show("Game Over!", title);
        }
    };
    button.SetValue(Grid.ColumnProperty, column);
    button.SetValue(Grid.RowProperty, row);
    grid.Children.Add(button);
}
```

The `Add(...)` method is used to add a `Button` to a `Grid` to contain each part of the game and to check if the game has been won, a draw, to swap the players or if the game is over

Universal Windows Platform – Tic Tac Toe

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

```
private void Layout(ref Grid grid)
{
    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());
    }
    // Setup Board
    for (int row = 0; (row < size); row++)
    {
        for (int column = 0; (column < size); column++)
        {
            Add(ref grid, row, column);
            _board[row, column] = blank;
        }
    }
}
```

`Layout(...)` configures a `Grid` and sets up the layout of the game using the `Add(...)` method and `_board`

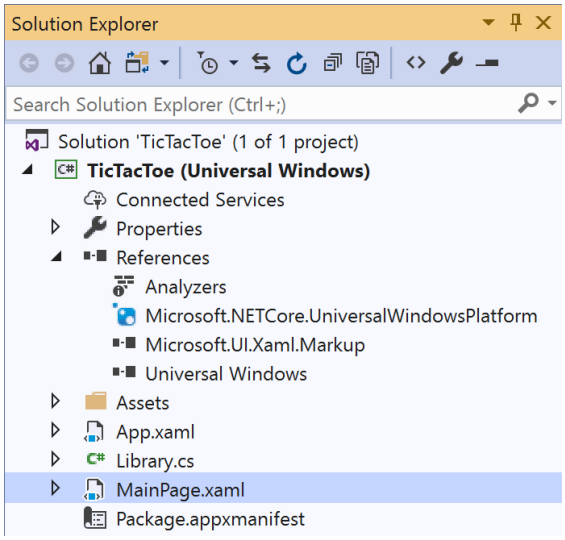
Finally after the **`private void Layout(...) { ... }` method** the following **`public` method** should be entered:

```
public async void New(Grid grid)
{
    Layout(ref grid);
    _won = false;
    _piece = await ConfirmAsync("Who goes First?", title,
        nought, cross) ? nought : cross;
}
```

`New(...)` will setup the layout of the `Grid` using the `Layout` method and will start the game and ask `Who goes First?` using the `ConfirmAsync(...)` method

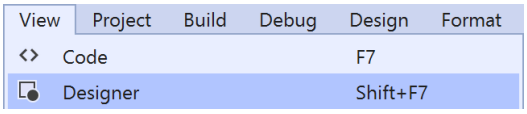
Universal Windows Platform – Tic Tac Toe

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**

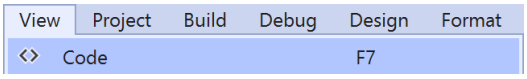
Step 7

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 a new game

Step 8



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

Universal Windows Platform – Tic Tac Toe

Step 9

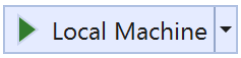
Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
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. In the **`New_Click(...)`** Event handler will call the **`New(...)`** method in the **`Library`** class

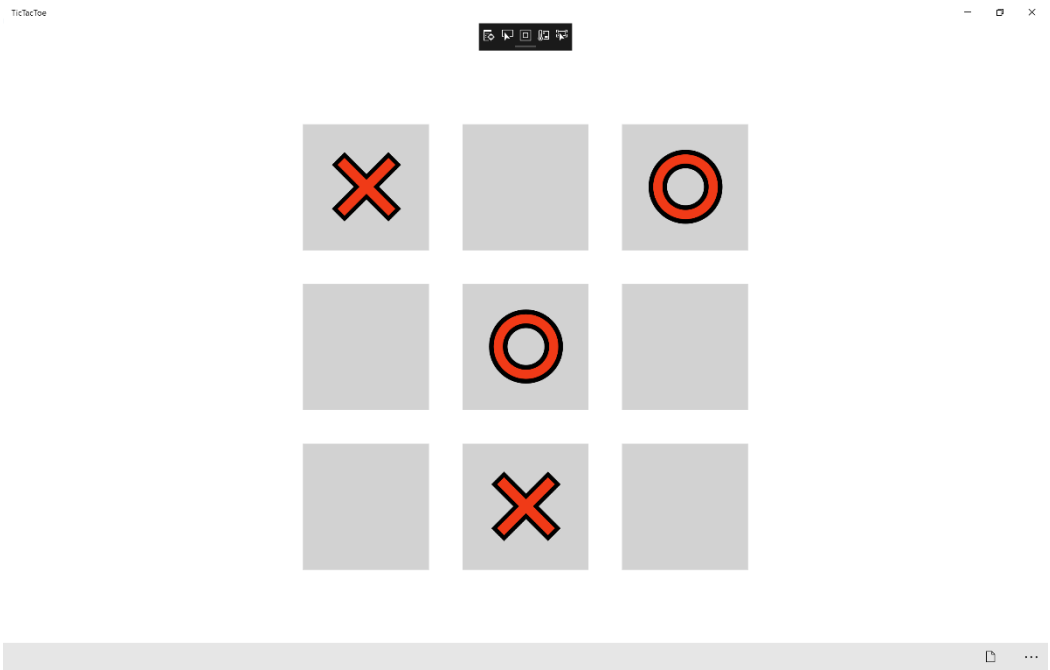
Step 10



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 you can then click the **New** Button, then choose **X** or **O** then you can start playing the game



Step 12

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