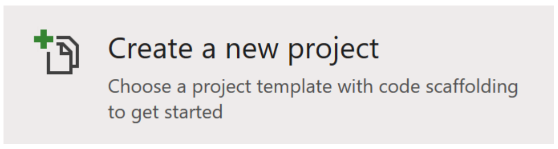
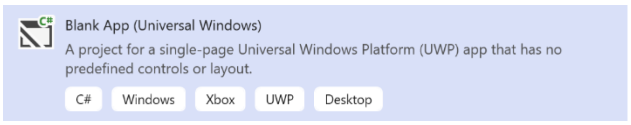
# Universal Windows Platform – Hit or Miss

**Hit or Miss** shows how to create a simple random game where you can score a **Hit** or a **Miss** based on which **Button** is clicked

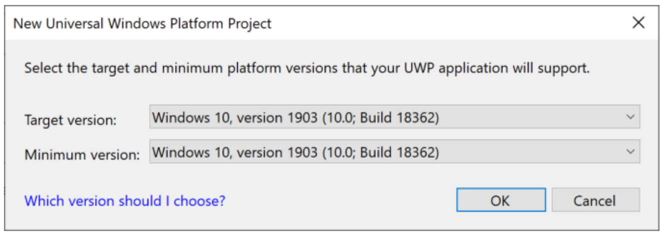
## Step 1


Create a new project
Choose a project template with code scaffolding to get started

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**


Blank App (Universal Windows)
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.
C#
Windows
Xbox
UWP
Desktop

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



New Universal Windows Platform Project

Select the target and minimum platform versions that your UWP application will support.

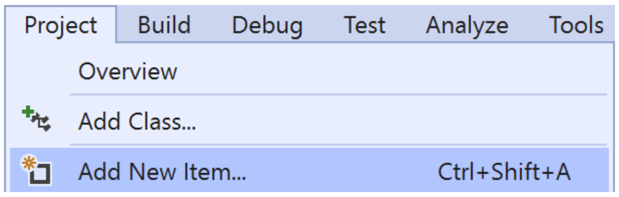
Target version: Windows 10, version 1903 (10.0; Build 18362)

Minimum version: Windows 10, version 1903 (10.0; Build 18362)

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**

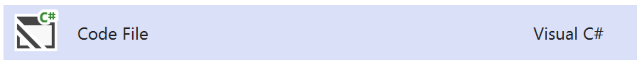
Target Version will control the most recent features of Windows 10 your application can use. To make sure you always have the most recent version, check for any Notifications or Updates in Visual Studio 2019

## 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 – Hit or Miss

## Step 4

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

```
using System;
using System.Collections.Generic;
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 = "Hit or Miss";
    private const string miss = "\U0001F573";
    private const string hit = "\U0001F4A5";
    private const int score = 18;
    private const int size = 6;

    private int _go = 0;
    private int _hits = 0;
    private int _misses = 0;
    private bool _won = false;
    private string[,] _board = new string[size, size];
    private Random _random = new Random((int)DateTime.Now.Ticks);


}
```

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

Then below the **`private Random _random = new Random((int)DateTime.UtcNow.Ticks);`** line the following **methods** should be entered:

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

private List<int> Choose(int minimum, int maximum, int total)
{
    int number;
    List<int> numbers = new List<int>();
    while (numbers.Count < total) // Select Numbers
    {
        number = _random.Next(minimum, maximum + 1);
        if (!numbers.Contains(number) || numbers.Count < 1)
        {
            numbers.Add(number); // Add if not Chosen or None
        }
    }
    return numbers;
}
```

The `Show` method is used to display a basic `MessageDialog` and `Choose` is used to return a `List<int>` of numbers using `Random`

Next below the **`private List<int> Choose(...) { ... }` method** the following **method** should be entered:

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

The `Piece` method is used to create a `TextBlock` for the `hit` or `miss` in the game

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

```
private void Add(ref Grid grid, int row, int column)
{
    Button button = new Button()
    {
        Width = 50,
        Height = 50,
        Margin = new Thickness(5),
        Style = (Style)Application.Current.Resources
            ["ButtonRevealStyle"]
    };
    button.Click += (object sender, RoutedEventArgs e) =>
    {
        if (!_won)
        {
            button = (Button)(sender);
            string selected = _board[(int)button.GetValue(Grid.RowProperty),
                (int)button.GetValue(Grid.ColumnProperty)];
            if (button.Content == null)
            {
                button.Content = (Piece(selected));
                if (selected == hit)
                    _hits++;
                else if (selected == miss)
                    _misses++;
                _go++;
            }
            if (_go < (size * size) && _misses < score)
            {
                if (_hits == score)
                {
                    Show($"You Won! With {_hits} hits and {_misses} misses",
                    title);
                    _won = true;
                }
            }
            else
            {
                Show($"You Lost! With {_hits} hits and {_misses} misses",
                title);
                _won = true;
            }
        }
    };
    button.SetValue(Grid.ColumnProperty, column);
    button.SetValue(Grid.RowProperty, row);
    grid.Children.Add(button);
}
```

The `Add` method is used to create the elements that will make up the game and will also check `if` have scored a `hit` or a `miss` and will also check if the game has been completed and if you won or lost

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

```
private void Layout(ref Grid grid)
{
    _go = 0;
    _hits = 0;
    _misses = 0;
    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());
    }
    for (int row = 0; (row < size); row++)
    {
        for (int column = 0; (column < size); column++)
        {
            Add(ref grid, row, column);
        }
    }
}
```

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

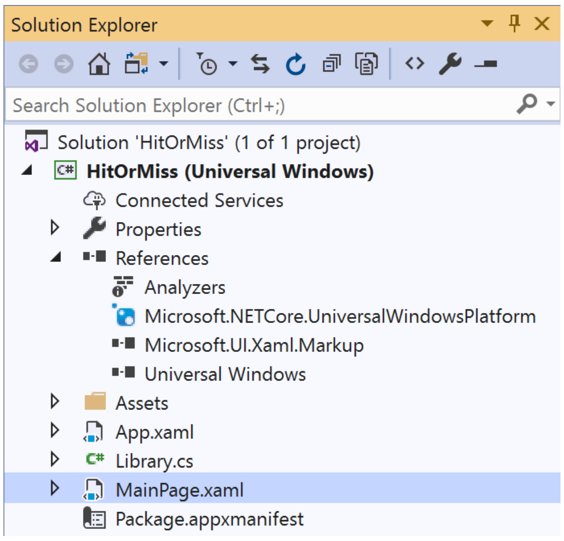
## Universal Windows Platform – Hit or Miss

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

```
public void New(ref Grid grid)
{
    Layout(ref grid);
    _won = false;
    int index = 0;
    // Setup Values
    List<string> values = new List<string>();
    while (values.Count < (size * size))
    {
        values.Add(hit);
        values.Add(miss);
    }
    List<int> indices = Choose(1, (size * size), (size * size));
    // Setup Board
    for (int column = 0; (column < size); column++)
    {
        for (int row = 0; (row < size); row++)
        {
            _board[column, row] = values[indices[index] - 1];
            index++;
        }
    }
}
```

The `New` method will setup the `values` for the game and will also setup the layout of the `Grid` using the `Layout` method

### Step 5


Solution Explorer

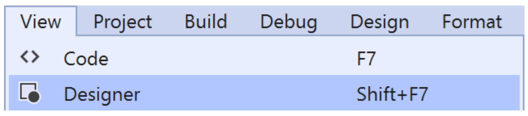
Search Solution Explorer (Ctrl+;)

Solution 'HitOrMiss' (1 of 1 project)
HitOrMiss (Universal Windows)
Connected Services
Properties
References
Analyzers
Microsoft.NETCore.UniversalWindowsPlatform
Microsoft.UI.Xaml.Markup
Universal Windows
Assets
App.xaml
Library.cs
MainPage.xaml
Package.appxmanifest

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

# Universal Windows Platform – Hit or Miss

## Step 6



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

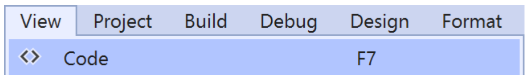
## Step 7

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**

## Step 9

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

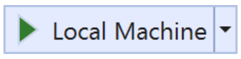
```
Library library = new Library();

private void New_Click(object sender, RoutedEventArgs e)
{
    library.New(ref 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

# Universal Windows Platform – Hit or Miss

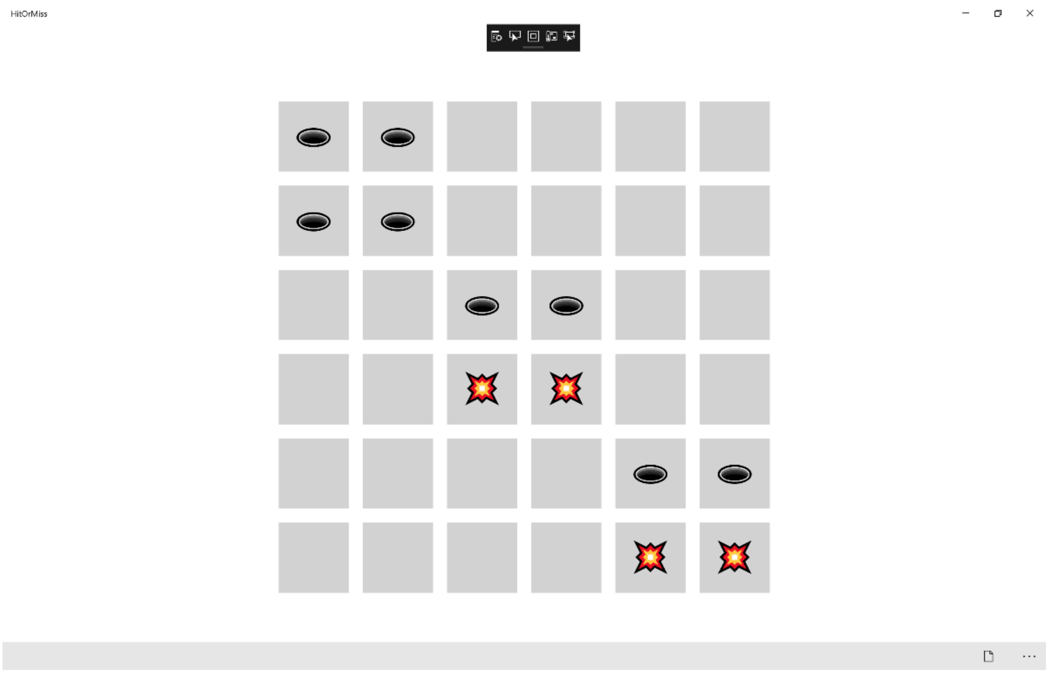
## Step 10



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 click **New** to start the playing, to win need to get more hits (**Explosions**) than misses (**Holes**) up to a total of **18** to win!



## Step 12

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