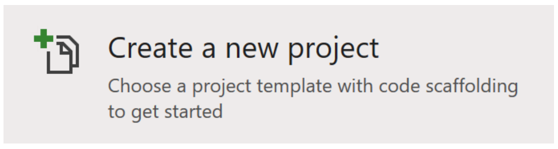
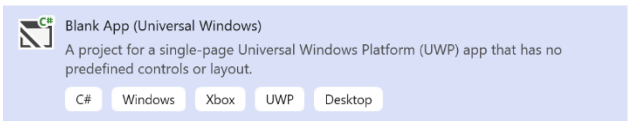
Universal Windows Platform – Light Game

Light Game shows how to create a simple game to toggle all the squares from **Gold** to **Black**

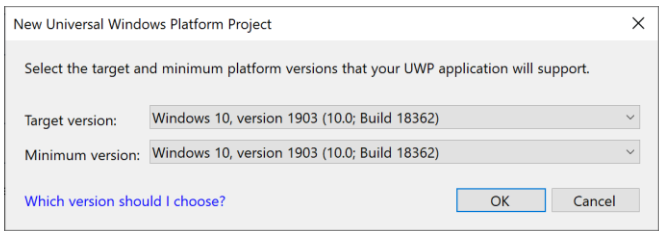
Step 1


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 **LightGame** and select **Create**



New Universal Windows Platform Project

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

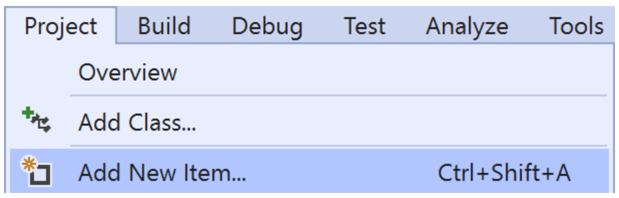
Target version: Windows 10, version 1903 (10.0; Build 18362)

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**

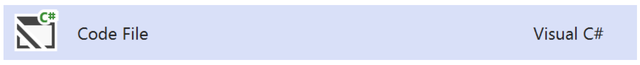
Target Version will control the most recent features of Windows 10 your application can use. To make sure you always have the most recent version, check for any Notifications or Updates in Visual Studio 2019

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 – Light Game

Step 4

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

```
using Windows.UI;
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 = "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 int _moves = 0;
    private bool _won = false;
    private int[,] _board = new int[size, size];


}
```

There are `using` statements to include necessary functionality. `_board` is a `int[,]` represents the overall state of the game and there are `Color` items for the on and off state colours

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 void Toggle(Grid grid, int row, int column)
{
    _board[row, column] = _board[row, column] == on ? off : on;
    Button element = (Button)grid.FindName($"{row}:{column}");
    element.Background = _board[row, column] == on ?
        new SolidColorBrush(lightOn) :
        new SolidColorBrush(lightOff);
}
```

`Show` method is used to display a `MessageDialog` and `Toggle` is used to set the `_board` states and set the `Background` property

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

```
private void Set(Grid grid, int row, int column)
{
    Toggle(grid, row, column);
    if (row > 0)
    {
        Toggle(grid, row - 1, column); // Toggle Left
    }
    if (row < (size - 1))
    {
        Toggle(grid, row + 1, column); // Toggle Right
    }
    if (column > 0)
    {
        Toggle(grid, row, column - 1); // Toggle Above
    }
    if (column < (size - 1))
    {
        Toggle(grid, row, column + 1); // Toggle Below
    }
}
```

`Set` method is used to call the `Toggle` method for the selected item and those to the Left and Right plus those Above and Below

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

```
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;
}
```

`Winner` method is used to check if the board has been completed and the game is over

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

```
private void Select(Grid grid, Button button)
{
    if (!_won)
    {
        int row = (int)button.GetValue(Grid.RowProperty);
        int column = (int)button.GetValue(Grid.ColumnProperty);
        Set(grid, row, column);
        _moves++;
        if (Winner())
        {
            Show($"Well Done! You won in {_moves} moves!", title);
            _won = true;
        }
    }
    else
    {
        Show($"Game Over!", title);
    }
}
```

`Select` method is used when a Button has been selected and controls how the game is played

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

```
private void Add(Grid grid, int row, int column)
{
    Button button = new Button()
    {
        Width = 50,
        Height = 50,
        Name = $"{row}:{column}",
        Margin = new Thickness(2),
        Background = new SolidColorBrush(lightOn),
        Style = (Style)Application.Current.Resources
        ["ButtonRevealStyle"]
    };
    button.Click += (object sender, RoutedEventArgs e) =>
    {
        Select(grid, button);
    };
    button.SetValue(Grid.ColumnProperty, column);
    button.SetValue(Grid.RowProperty, row);
    grid.Children.Add(button);
}
```

`Add` method is used to create a `Button` and call the `Select` method when it is clicked

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

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

`Layout` method is used to create the look-and-feel of the game and the `Grid` and call the `Add` method

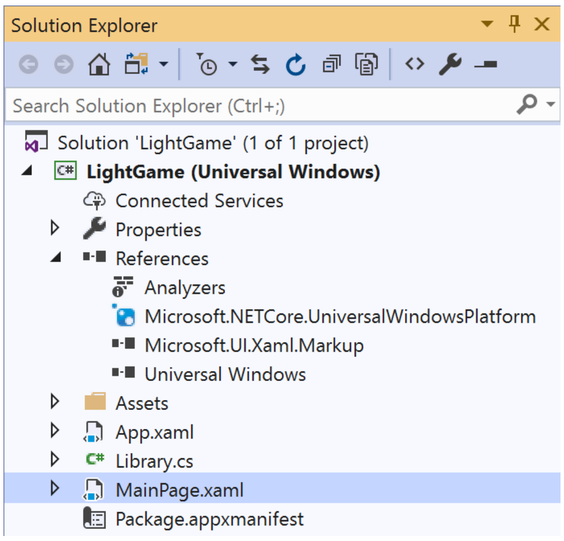
Finally after **`private void Layout(...) { ... }` method** the following **`public` method** should be entered:

```
public void New(Grid grid)
{
    _moves = 0;
    _won = false;
    Layout(grid);
    // Reset Board
    for (int column = 0; (column < size); column++)
    {
        for (int row = 0; (row < size); row++)
        {
            _board[column, row] = on;
        }
    }
}
```

`New` method will setup and start playing the game by calling `Layout` method

Universal Windows Platform – Light Game

Step 5

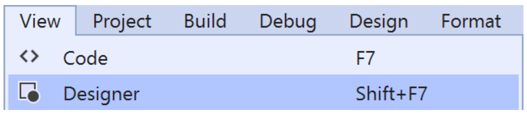

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'LightGame' (1 of 1 project)
LightGame (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**

Step 6



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

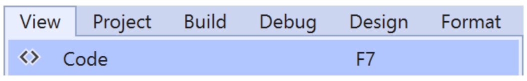
Step 7

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 Name="Display" Margin="50"
    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 Grid to represent the game and the second block of XAML is the CommandBar which contains New to setup and start the game

Step 8



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

Universal Windows Platform – Light Game

Step 9

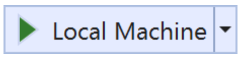
Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
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 setup and play the game using the `New` method in the `Library` class

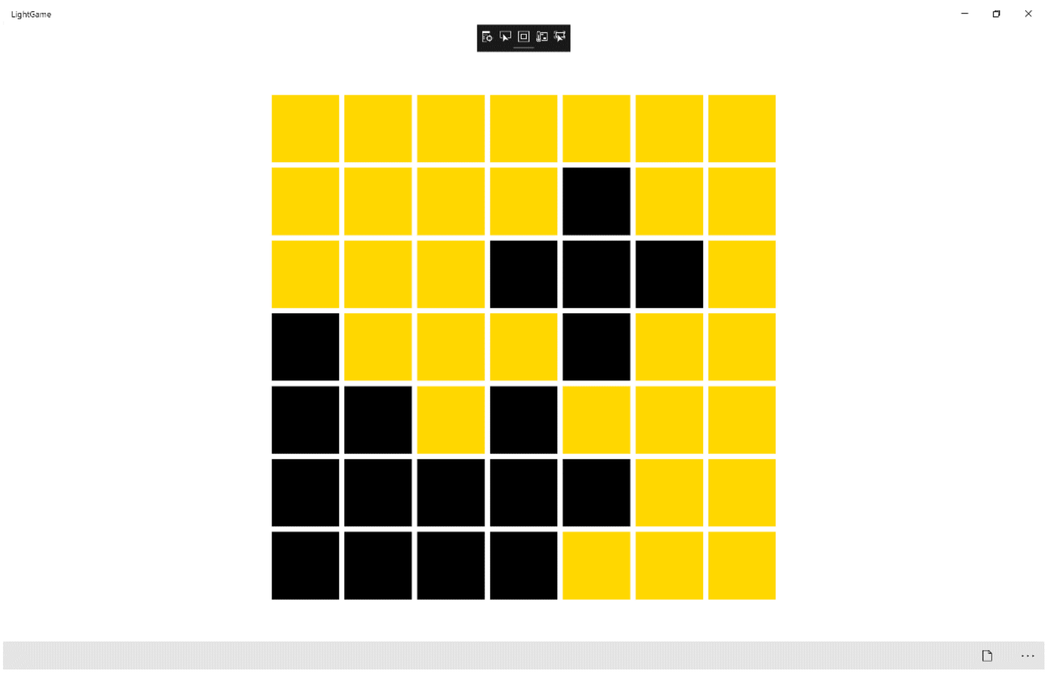
Step 10



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 playing, you can win by setting all the **Gold** squares to **Black**



Step 12

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