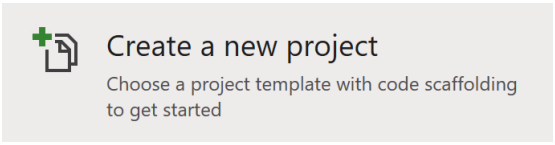
Universal Windows Platform – Lucky Bingo

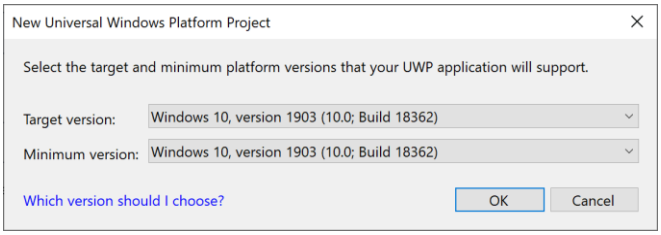
Lucky Bingo shows how to create a **Bingo** game using **Grid** and **Ellipse** controls to display which balls have been drawn and which match a random ticket until you get a **Full House**

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**

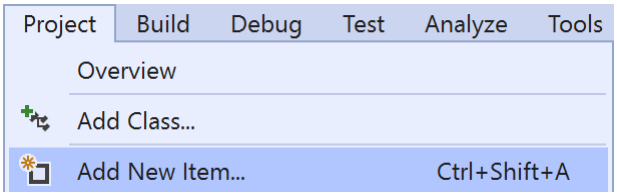
Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **LuckyBingo** and select **Create**



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 – Lucky Bingo

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 System.Linq;
using Windows.UI;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Shapes;

public class Library
{
    private const string title = "Lucky Bingo";

    private const int size = 22;
    private const int balls = 90;
    private const int marks = 25;
    private const int maximum = 90;
    private readonly Color _accent =
        (Color)Application.Current.Resources["SystemAccentColor"];

    private int _count;
    private int _house;
    private List<int> _balls;
    private List<int> _marks;
    private bool _gameOver = false;
    private Random _random = new Random((int)DateTime.UtcNow.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 total)
{
    int number;
    List<int> numbers = new List<int>();
    while (numbers.Count < total) // Select Numbers
    {
        number = _random.Next(1, 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(int total) { ... }` method** the following **method** should be entered:

```
private Ellipse Ellipse(bool ball, int value)
{
    Ellipse ellipse = new Ellipse()
    {
        Width = size,
        Height = size,
        VerticalAlignment = VerticalAlignment.Center,
        HorizontalAlignment = HorizontalAlignment.Center
    };
    if (ball)
    {
        ellipse.StrokeThickness = 2;
        ellipse.Stroke = new SolidColorBrush(_accent);
    }
    else
    {
        ellipse.Opacity = 0;
        ellipse.Name = $"mark{value}";
        ellipse.Fill = new SolidColorBrush(_accent);
    }
    return ellipse;
}
```

The `Ellipse` method is used to create an `Ellipse` for the `ball` or `else` for indicating a `mark` on the ticket

Then after the **`private Grid Pocket(int value) { ... }` method** the following **method** should be entered:

```
private void Add(ref Grid grid, bool ball,
    int row, int column, int value)
{
    Grid element = new Grid()
    {
        Width = size,
        Height = size,
    };
    if (ball)
    {
        element.Opacity = 0;
        element.Name = $"ball{value}";
    }
    TextBlock label = new TextBlock()
    {
        FontSize = 12,
        Text = $"{value}",
        TextLineBounds = TextLineBounds.Tight,
        VerticalAlignment = VerticalAlignment.Center,
        Foreground = new SolidColorBrush(Colors.Black),
        HorizontalAlignment = HorizontalAlignment.Center,
    };
    element.Children.Add(label);
    element.Children.Add(Ellipse(ball, value));
    element.SetValue(Grid.RowProperty, row);
    element.SetValue(Grid.ColumnProperty, column);
    grid.Children.Add(element);
}
```

The `Add` method is used to create the elements that will make up a `ball` or that make up a ticket

Universal Windows Platform – Lucky Bingo

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

```
private Grid Layout(bool ball, int rows,
    int cols, List<int> list)
{
    int count = 0;
    Grid grid = new Grid()
    {
        Width = 250,
        Height = 250
    };
    grid.Children.Clear();
    grid.RowDefinitions.Clear();
    grid.ColumnDefinitions.Clear();
    // Setup Grid
    for (int row = 0; row < rows; row++)
    {
        grid.RowDefinitions.Add(new RowDefinition());
    }
    for (int column = 0; column < cols; column++)
    {
        grid.ColumnDefinitions.Add(new ColumnDefinition());
    }
    // Setup Board
    for (int row = 0; row < rows; row++)
    {
        for (int column = 0; column < cols; column++)
        {
            Add(ref grid, ball, row, column, list[count]);
            count++;
        }
    }
    return grid;
}
```

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

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

```
private void Set(ref StackPanel panel, bool isBall,
int value, double opacity)
{
    string name = $"{(isBall ? "ball" : "mark")}{value}";
    UIElement element = (UIElement)panel.FindName(name);
    if (element != null) element.Opacity = opacity;
}
```

The `Set` method will get a `UIElement` using `FindName` for the drawn `ball` or `mark` a matched value

Universal Windows Platform – Lucky Bingo

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

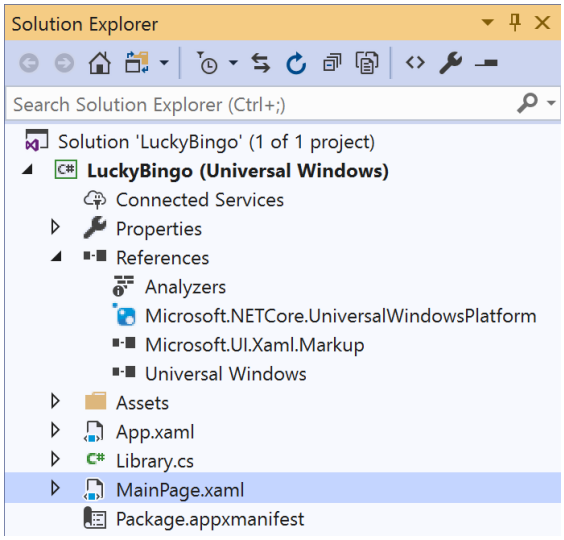
```
public void New(StackPanel panel)
{
    _count = 0;
    _house = 0;
    _gameOver = false;
    _balls = Choose(balls);
    _marks = Choose(marks);
    panel.Children.Clear();
    panel.Children.Add(Layout(true, 9, 10, _balls));
    panel.Children.Add(Layout(false, 5, 5, _marks));
}

public void Play(StackPanel panel)
{
    if (!panel.Children.Any()) New(panel);
    if (_count < balls && !_gameOver)
    {
        var ball = _balls[_count];
        Set(ref panel, true, ball, 1);
        if (_marks.Contains(ball))
        {
            _house++;
            Set(ref panel, false, ball, 0.5);
            if (_house == marks)
            {
                _gameOver = true;
                Show($"Full House in {_count} Balls!", title);
            }
        }
        _count++;
    }
    else
    {
        Show($"Game Over!", title);
    }
}
```

The `New` method will setup the layout of the `Grid` using the `Layout` method using `Choose` the values for the ticket with `marks` and numbers to be drawn with `balls` and the `Play` method will check to see if a drawn value is the same as one in the ticket and when the `_house` value reaches the total in the ticket that's a full house and the game is won, or if not and all numbers are drawn then the game will be over

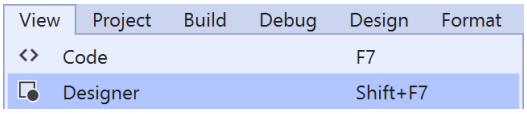
Universal Windows Platform – Lucky Bingo

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

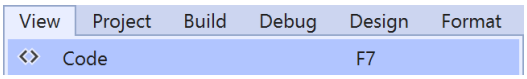
In the **Design** View and **XAML** View of **Visual Studio 2019** will be displayed, and in this between the **Grid** and **/Grid** elements enter the following **XAML**:

```
<Viewbox>
    <StackPanel Margin="50" Name="Display"
    Orientation="Horizontal"
    HorizontalAlignment="Center"
    VerticalAlignment="Center"/>
</Viewbox>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Icon="Page2" Label="New" Click="New_Click"/>
    <AppBarButton Icon="Play" Label="Play" Click="Play_Click"/>
</CommandBar>
```

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

Universal Windows Platform – Lucky Bingo

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(Display);
}

private void Play_Click(object sender, RoutedEventArgs e)
{
    library.Play(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 and the `Play_Click` event handler will call the `Play` method

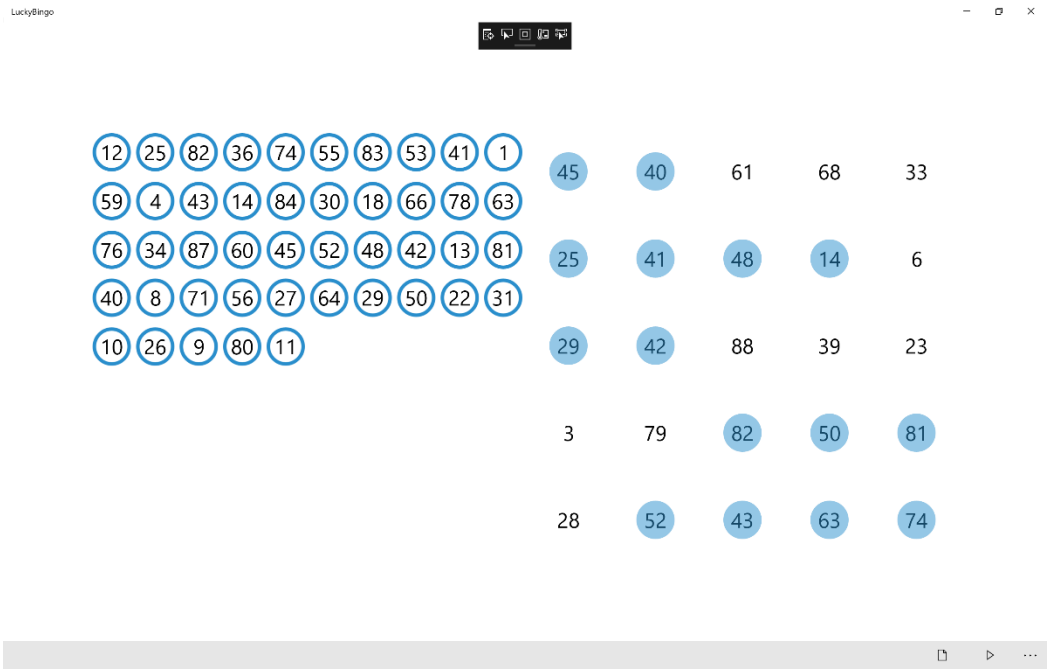
Universal Windows Platform – Lucky Bingo

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 setup the **Bingo Game** then click **Play** which will draw a **Ball** and if this matches the ticket it will be marked off, mark off all the numbers for a **Full House**



You can also add more functionality to this game to support 1-line, 2-line and other prizes for a more functional bingo game

Step 12

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