**Memory Game** shows how to use **Grid** to implement a simple memory game to pair moon phases

## 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** |
| A screenshot of a cell phone  Description automatically generated | Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **MemoryGame** and select **Create** |
| A screenshot of a social media post  Description automatically generated | 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

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated | Choose **Project** then **Add New Item...** from the **Menu** in **Visual Studio 2019** |

## Step 3

|  |  |
| --- | --- |
| A close up of a logo  Description automatically generated | Then choose **Code File** from **Add New Item** in **Visual Studio 2019**, enter the **Name** as **Library.cs** and select **Add** |

## 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.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 = "Memory Game";  private const int size = 4;  private int \_moves = 0;  private int \_firstId = 0;  private int \_secondId = 0;  private Button \_first;  private Button \_second;  private int[,] \_board = new int[size, size];  private List<int> \_matches = new List<int>();  private Random \_random = new Random((int)DateTime.Now.Ticks);  private Dictionary<int, string> \_moon = new Dictionary<int, string>()  {  { 1, "\U0001F311" }, // New  { 2, "\U0001F312" }, // Waxing Crescent  { 3, "\U0001F313" }, // First Quarter  { 4, "\U0001F314" }, // Waxing Gibbous  { 5, "\U0001F315" }, // Full  { 6, "\U0001F316" }, // Waning Gibbous  { 7, "\U0001F317" }, // Last Quarter  { 8, "\U0001F318" } // Waning Crescent  };    } |

There are using statements to include necessary functionality. There are private **members** for various parts of the game including the two-dimensional array \_board which represents what will appear in the game and a Dictionary<int, string> to represent the **Emoji** characters to represent each phase of the moon that should be matched

Then below the **private Dictionary<int, string> \_moon = new Dictionary<int, string>() { ... };** the following **methods** should be entered:

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

Show() is used to display a basic MessageDialog

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

|  |
| --- |
| private Viewbox Phase(int value)  {  TextBlock textblock = new TextBlock()  {  Text = \_moon[value],  IsColorFontEnabled = true,  TextLineBounds = TextLineBounds.Tight,  FontFamily = new FontFamily("Segoe UI Emoji"),  HorizontalTextAlignment = TextAlignment.Center  };  return new Viewbox()  {  Child = textblock  };  } |

Phase(...) is used to return a Viewbox which contains a TextBlock which will contain the **Emoji** value of a phase of the moon

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

|  |
| --- |
| private List<int> Choose(int start, int maximum, int total)  {  int number;  List<int> numbers = new List<int>();  while ((numbers.Count < total)) // Select Numbers  {  // Random Number between Start and Finish  number = \_random.Next(start, maximum + 1);  if ((!numbers.Contains(number)) || (numbers.Count < 1))  {  numbers.Add(number); // Add if number Chosen or None  }  }  return numbers;  } |

Choose() **method** is used to return a List<int> of numbers using Random to select them

Then after the **private List<int> Choose(...) { ... }** **method** the following **methods** should be entered:

|  |
| --- |
| private void Match()  {  \_matches.Add(\_firstId);  \_matches.Add(\_secondId);  if (\_first != null)  {  \_first.Background = null;  \_first = null;  }  if (\_second != null)  {  \_second.Background = null;  \_second = null;  }  if (\_matches.Count == size \* size)  {  Show($"Matched all moon phases in {\_moves} moves!", title);  }  }  private async void NoMatch()  {  await Task.Delay(TimeSpan.FromSeconds(1.5));  if (\_first != null)  {  \_first.Content = null;  \_first = null;  }  if (\_second != null)  {  \_second.Content = null;  \_second = null;  };  }  private void Compare()  {  if (\_firstId == \_secondId)  Match();  else  NoMatch();  \_moves++;  \_firstId = 0;  \_secondId = 0;  } |

Match() **method** will handle what should happen when a pair of phases patch and then if the game is over will display a message. NoMatch() **method** will reset selected items after a Delay of **1.5** seconds and the Compare **method** will handle when to call those **methods**

Next after the **private void Compare()** **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) =>  {  int selected;  button = (Button)(sender);  row = (int)button.GetValue(Grid.RowProperty);  column = (int)button.GetValue(Grid.ColumnProperty);  selected = \_board[row, column];  if ((\_matches.IndexOf(selected) < 0))  {  if (\_firstId == 0) // No Match  {  \_first = button;  \_firstId = selected;  \_first.Content = Phase(selected);  }  else if (\_secondId == 0)  {  \_second = button;  if (!\_first.Equals(\_second)) // Different  {  \_secondId = selected;  \_second.Content = Phase(selected);  Compare();  }  }  }  };  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 it will be used to check if the \_first item selected with a Button and calls the Compare() **method**

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

|  |
| --- |
| private void Layout(ref Grid grid)  {  \_moves = 0;  \_matches.Clear();  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);  }  }  } |

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 void New(Grid grid)  {  Layout(ref grid);  int counter = 0;  List<int> values = new List<int>();  // Pairs : Random 1 - 8  while (values.Count <= size \* size)  {  List<int> numbers = Choose(1, size \* 2, size \* 2);  for (int number = 0; number < size \* 2; number++)  {  values.Add(numbers[number]);  }  }  // Board : Random 1 - 16  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[counter] - 1];  counter++;  }  }  } |

New(...) will setup the layout of the Grid using the Layout **method**, it will also select the **Pairs** of items that will be matched in the game and setup the **Board**

## Step 5

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

## Step 6

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated | 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** |

## 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);  } |

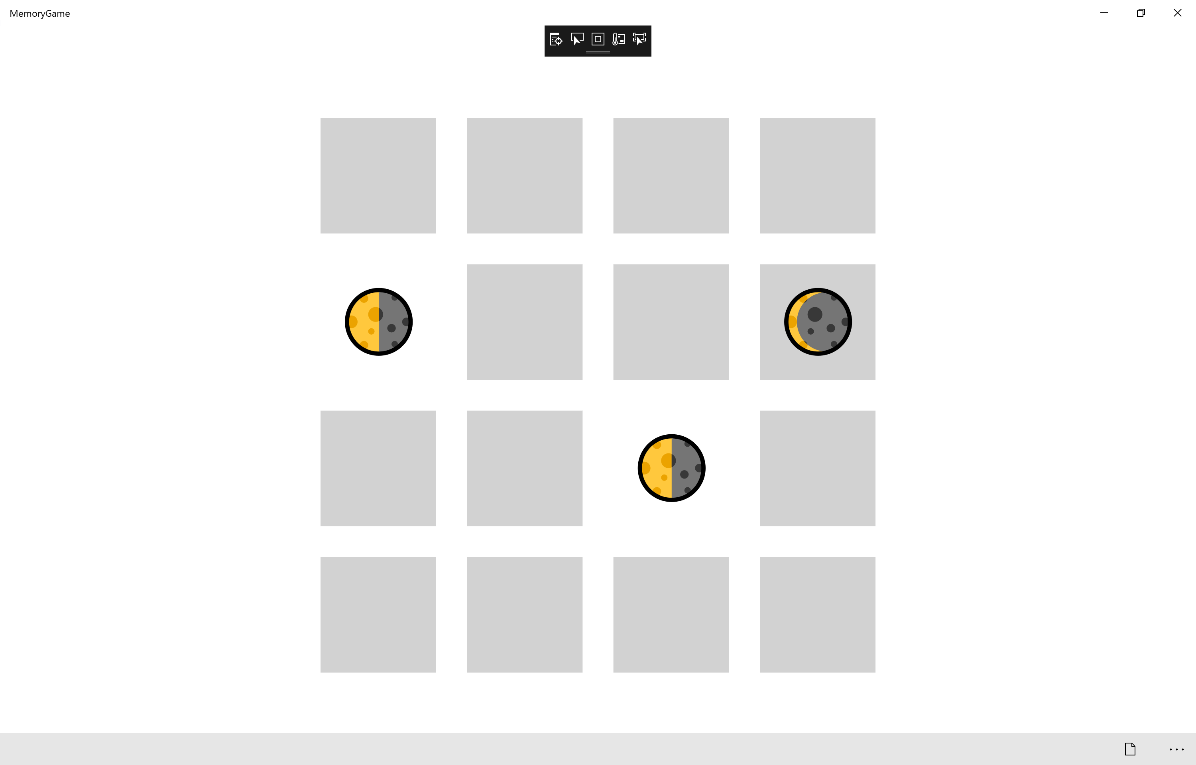
Below the **MainPage(...)** method an instance of the Library **Class** is created. In the New\_Click(...) **Event** handler will call the New(...) **method** in the Library **class**

## 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 click the **New** Button and then click on any two **Buttons** to display a phase of the moon, match the phases to make a pair and match them all to win!



## Step 12

|  |  |
| --- | --- |
| A picture containing object  Description automatically generated | To Exit the Application, select the **Close** button in the top right of the Application |