**Lucky Dominoes** shows how to create the look-and-feel of a **Domino** as well as obeying the rules of one and could for the basis of any **Domino** based game

## 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 **LuckyDominoes** 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.Linq;  using Windows.UI;  using Windows.UI.Xaml;  using Windows.UI.Xaml.Controls;  using Windows.UI.Xaml.Media;  using Windows.UI.Xaml.Shapes;  public class Library  {  private const int size = 3;  private const string set\_one = "one";  private const string set\_two = "two";  private const string name\_upper = "upper";  private const string name\_lower = "lower";  private readonly string[] tiles =  {  "0,0",  "0,1", "1,1",  "0,2", "1,2", "2,2",  "0,3", "1,3", "2,3", "3,3",  "0,4", "1,4", "2,4", "3,4", "4,4",  "0,5", "1,5", "2,5", "3,5", "4,5", "5,5",  "0,6", "1,6", "2,6", "3,6", "4,6", "5,6", "6,6"  };  private readonly byte[][] layout =  {  // a, b, c, d, e, f, g, h, i  new byte[] { 0, 0, 0, 0, 0, 0, 0, 0, 0 }, // 0  new byte[] { 0, 0, 0, 0, 1, 0, 0, 0, 0 }, // 1  new byte[] { 1, 0, 0, 0, 0, 0, 0, 0, 1 }, // 2  new byte[] { 1, 0, 0, 0, 1, 0, 0, 0, 1 }, // 3  new byte[] { 1, 0, 1, 0, 0, 0, 1, 0, 1 }, // 4  new byte[] { 1, 0, 1, 0, 1, 0, 1, 0, 1 }, // 5  new byte[] { 1, 0, 1, 1, 0, 1, 1, 0, 1 }, // 6  };  private readonly string[] tags =  {  "a", "b", "c", "d", "e", "f", "g", "h", "i"  };  private int \_turns = 0;  private List<int> \_one = new List<int>();  private List<int> \_two = new List<int>();  private Random \_random = new Random((int)DateTime.Now.Ticks);    } |

There are using statements to include necessary functionality. layout is a byte[][] is a **two-dimensional array** of values that will represent which row and column of pips will be displayed on the **upper** or **lower** portion of a **domino** and Random is used to create the numbers for the **domino**

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

|  |
| --- |
| private Color Colour(string resource)  {  return (Color)Application.Current.Resources[resource];  }  private Brush Background()  {  return new LinearGradientBrush(new GradientStopCollection()  {  new GradientStop()  {  Color = Colour("SystemAccentColorLight3"),  Offset = 0.0  },  new GradientStop()  {  Color = Colour("SystemAccentColorDark3"),  Offset = 1.0  }  }, 90);  } |

Colour **method** is used to return Color **resource** and Background **method** is used to create a LinearGradientBrush using that **method**

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

|  |
| --- |
| private List<int> Choose(int total)  {  return Enumerable.Range(0, total)  .OrderBy(r => \_random.Next(0, total)).ToList();  }  private void Add(Grid grid, int row, int column, string name)  {  Ellipse element = new Ellipse()  {  Name = name,  Opacity = 0,  Margin = new Thickness(5),  Fill = new SolidColorBrush(Colors.WhiteSmoke)  };  element.SetValue(Grid.ColumnProperty, column);  element.SetValue(Grid.RowProperty, row);  grid.Children.Add(element);  } |

Choose **method** is used to get a randomised List of int using Enumerable and Add **method** is used to create the Ellipse to represent the pips of the **domino**

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

|  |
| --- |
| private Grid Portion(string name)  {  Grid grid = new Grid()  {  Name = name,  Width = 100,  Height = 100,  Background = Background(),  Padding = new Thickness(5),  };  // Setup Grid  for (int index = 0; (index < size); index++)  {  grid.RowDefinitions.Add(new RowDefinition());  grid.ColumnDefinitions.Add(new ColumnDefinition());  }  int count = 0;  // Setup Layout  for (int row = 0; (row < size); row++)  {  for (int column = 0; (column < size); column++)  {  Add(grid, row, column, $"{name}.{tags[count]}");  count++;  }  }  return grid;  } |

Portion **method** is used to create the layout of part of a **domino** by using the Add **method**

Then after the **private Grid Portion(...) { ... }** **method** the following **methods** should be entered:

|  |
| --- |
| private StackPanel Domino(string name)  {  StackPanel panel = new StackPanel()  {  Margin = new Thickness(25),  Orientation = Orientation.Vertical  };  panel.Children.Add(Portion($"{name}.{name\_upper}"));  panel.Children.Add(Portion($"{name}.{name\_lower}"));  return panel;  }  private void Layout(Grid grid)  {  StackPanel panel = new StackPanel()  {  Orientation = Orientation.Horizontal  };  panel.Children.Add(Domino(set\_one));  panel.Children.Add(Domino(set\_two));  grid.Children.Add(panel);  } |

Domino **method** uses the Portion **method** to create the layout of a **domino** using a StackPanel and then is used by Layout **method** to create a pair of dominoes on another StackPanel

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

|  |
| --- |
| private void SetPortion(Grid grid, string name, int value)  {  for (int i = 0; i < tags.Length; i++)  {  ((UIElement)grid.FindName($"{name}.{tags[i]}"))  .Opacity = layout[value][i];  }  }  private void Set(Grid grid, string name, string tile)  {  string[] pair = tile.Split(',');  SetPortion(grid, $"{name}.{name\_upper}", int.Parse(pair[0]));  SetPortion(grid, $"{name}.{name\_lower}", int.Parse(pair[1]));  } |

SetPortion **method** is used to set the Opacity of an element and Set **method** is used to update the value of a **domino** using the SetPortion **method**

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

|  |
| --- |
| public void New(Grid grid)  {  Layout(grid);  \_turns = tiles.Count() - 1;  \_one = Choose(tiles.Count());  \_two = Choose(tiles.Count());  }  public void Play(Grid grid)  {  if (!grid.Children.Any()) New(grid);  if (\_turns > 0)  {  Set(grid, set\_one, tiles[\_one[\_turns]]);  Set(grid, set\_two, tiles[\_two[\_turns]]);  \_turns--;  }  else  {  New(grid);  }  } |

New **method** will setup the layout of the Grid using the Layout **method** and select the randomised values to display on a **domino** and Play **method** will use the Set **method** to display the value of a **domino** until there's no more to display

## 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"/>  <AppBarButton Icon="Play" Label="Play" Click="Play\_Click"/>  </CommandBar> |

The first block of **XAML** the main user interface features a **Grid** with two **Grid** Controls within to represent the **dominoes**. The second block of **XAML** is the **CommandBar** which contains **New** to reset the game and **Play** to show the **dominoes**

## 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);  } |

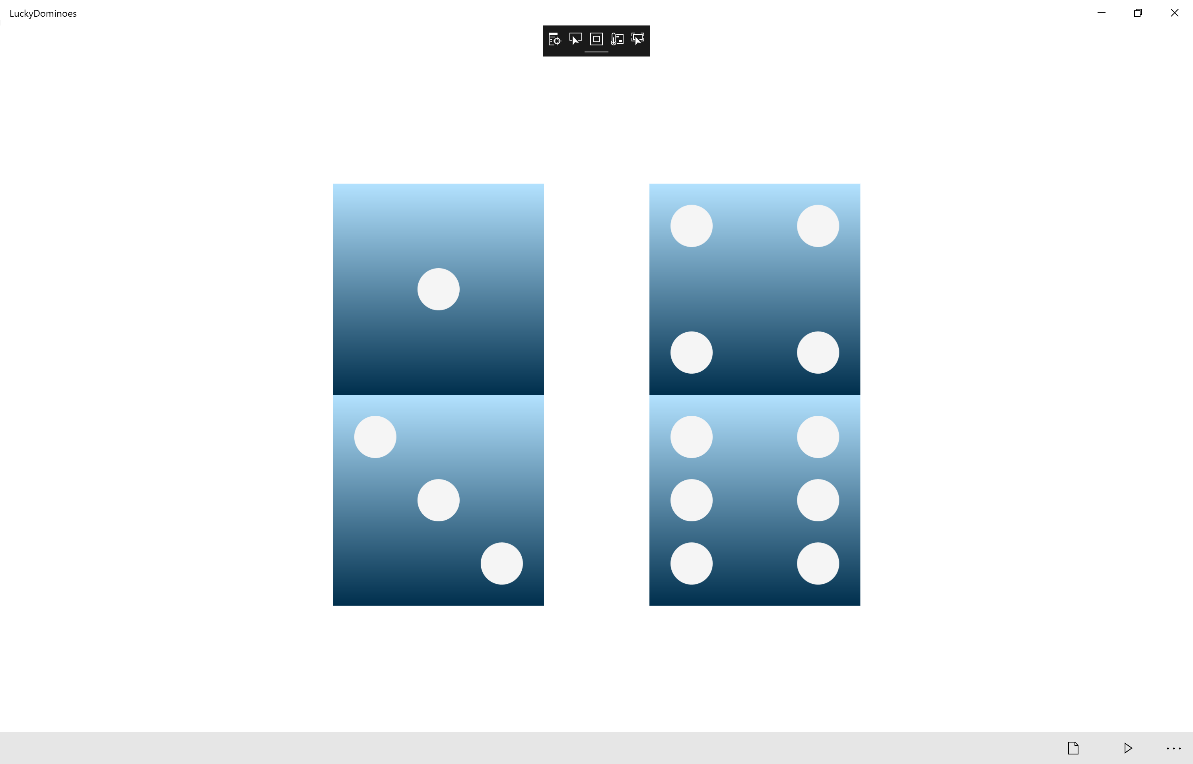
Below the **MainPage** method an instance of the Library **class** is created. In the New\_Click(...) **Event** handler will setup the game with the New **method**, and Play\_Click(...) will call the Play **method** in the Library **class**

## 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 use **New** to start then **Play** to set each **Domino** to a random selection of all the possible values of a **Domino**



## Step 12

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