**Touch Game** shows how to use a **Grid** to implement a Touch-based pattern matching 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 **TouchGame** 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.Popups;  using Windows.UI.Xaml;  using Windows.UI.Xaml.Controls;  using Windows.UI.Xaml.Media;  public class Library  {  private const string title = "Touch Game";  private const int size = 2;  private const int speed = 800;  private const int light = 400;  private const int click = 200;  private const int level = 100;  private readonly Dictionary<int, string> \_square =  new Dictionary<int, string>()  {  { 0, "\U0001F7E5" }, // Red Square  { 1, "\U0001F7E6" }, // Blue Square  { 2, "\U0001F7E9" }, // Green Square  { 3, "\U0001F7E8" }, // Yellow Square  };  private int \_turn = 0;  private int \_count = 0;  private bool \_play = false;  private bool \_isTimer = false;  private List<int> \_items = new List<int>();  private DispatcherTimer \_timer = new DispatcherTimer();  private Random \_random = new Random((int)DateTime.Now.Ticks);  } |

There are using statements to include necessary functionality. \_suits is a Dictionary<int, string> represents the different coloured squares that will be shown and Random is used to create the numbers for the game

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

|  |
| --- |
| public void Show(string content, string title)  {  \_ = new MessageDialog(content, title).ShowAsync();  }  private List<int> Choose(int start, int finish, int total)  {  int number;  List<int> numbers = new List<int>();  while (numbers.Count < total) // Select Numbers  {  // Random non-unique Number between Start and Finish  number = \_random.Next(start, finish + 1);  numbers.Add(number); // Add Number  }  return numbers;  } |

Show **method** is used to display a basic MessageDialog and Choose **method** is use to pick a set of randomised numbers

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

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

Square **method** is used to create a TextBlock which be used for each of the **squares** of the game

After the **private Viewbox Square(...) { ... }** **method** the following **method** should be entered:

|  |
| --- |
| private void Score(int value)  {  if (value == \_items[\_count])  {  if (\_count < \_turn)  {  \_count++;  }  else  {  \_isTimer = true;  \_play = false;  \_count = 0;  \_turn++;  }  }  else  {  Show($"Game Over! You scored {\_turn}!", title);  \_isTimer = false;  \_play = false;  \_count = 0;  \_turn = 0;  \_timer.Stop();  }  } |

Score **method** is used to work out the **score** for the game and also set other **member** values

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

|  |
| --- |
| private void Set(Grid grid, int value, int period)  {  Button button = (Button)grid.Children.Single(s =>  (int)((Button)s).Tag == value);  button.Opacity = 0.25;  DispatcherTimer opacity = new DispatcherTimer()  {  Interval = TimeSpan.FromMilliseconds(period)  };  opacity.Tick += (object sender, object e) =>  {  button.Opacity = 1.0;  opacity.Stop();  };  opacity.Start();  } |

Set **method** is used to setup the indication of a **square** of the game being selected

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

|  |
| --- |
| private void Tick(Grid grid)  {  if (\_isTimer)  {  if (\_count <= \_turn)  {  Set(grid, \_items[\_count], light);  \_count++;  }  if (\_count > \_turn)  {  \_isTimer = false;  \_play = true;  \_count = 0;  }  }  } |

Tick **method** is used to call the Set **method**

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

|  |
| --- |
| private void Add(Grid grid, int row, int column, int count)  {  Button button = new Button()  {  Tag = count,  Width = 100,  Height = 100,  Content = Square(count),  Margin = new Thickness(5)  };  button.Click += (object sender, RoutedEventArgs e) =>  {  if (\_play)  {  int value = (int)((Button)sender).Tag;  Set(grid, value, click);  Score(value);  }  };  button.SetValue(Grid.ColumnProperty, column);  button.SetValue(Grid.RowProperty, row);  grid.Children.Add(button);  } |

Add **method** will setup a Button and the Click **event** handler

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

|  |
| --- |
| private void Layout(ref Grid grid)  {  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());  }  int count = 0;  // Setup Board  for (int column = 0; (column < size); column++)  {  for (int row = 0; (row < size); row++)  {  Add(grid, row, column, count);  count++;  }  }  } |

Layout **method** will setup the layout of the game and the board

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

|  |
| --- |
| public void New(Grid grid)  {  Layout(ref grid);  \_items = Choose(0, 3, level);  \_play = false;  \_turn = 0;  \_count = 0;  \_isTimer = true;  \_timer = new DispatcherTimer  {  Interval = TimeSpan.FromMilliseconds(speed)  };  \_timer.Tick += (object sender, object e) =>  {  Tick(grid);  };  \_timer.Start();  } |

New **method** will setup and start playing the game by calling Layout and Choose methods

## 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 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** |

## 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);  } |

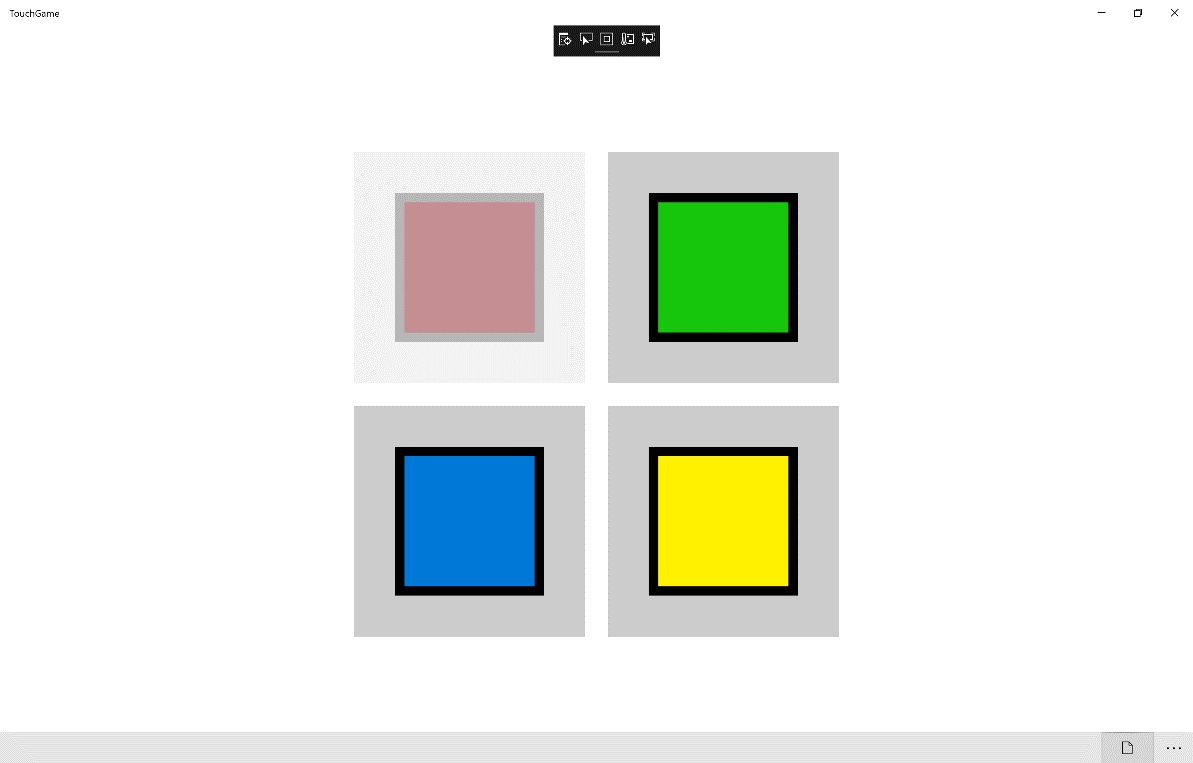
Below the **MainPage** method an instance of the Library **class** is created. In the New\_Click(...) **Event** handler will setup the game with the New **method** in the Library **class**

## 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 then click the **New** Button, then one of the squares will highlight, select the correct one, then each time one more square will highlight each turn, match the patterns to continue



## Step 12

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