Beispielprojekt: ClickMe

Inhaltsverzeichnis / list of contents

Das Spiel

Das Konzept:

Unser Projekt soll ein mittelgroßes Fenster sein, wo ein Button seine Position durchgehend zufällig ändert. Das Ziel ist es, auf den Button mit der Maus zu klicken, je mehr man klickt, desto schneller wird der Button. Nach einigen Klicks, soll der Button sein Maximum erreicht haben und man hat gewonnen. Dies müssen wir jetzt nur noch umsetzen.

Die graphische Oberfläche:

Nun kann es endlich losgehen.

Starte **Visual Studio**, gehe oben Links auf die Ecke und drücke auf **Datei Neu Projekt WPF-Anwendung**.

Du kannst nun das Projekt nach belieben benennen, oder speichern wo du willst. Falls du willst, kannst du auch selbst kreativ werden und das Programm designen, dennoch poste ich den **XAML-Code** rein.

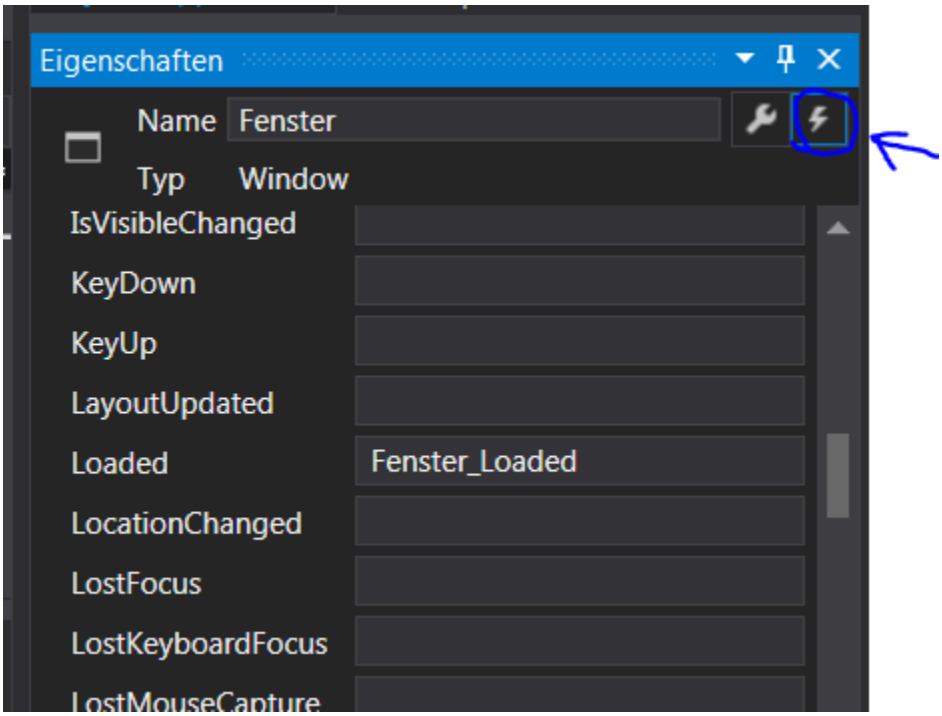
XAML-Code

```
<Window x:Name="clickme_window" x:Class="PROJEKTNAME_HIER_EINFUEGEN.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:PROJEKTNAME_HIER_EINFUEGEN"
        mc:Ignorable="d"
        Title="ClickMe" Height="623.5" Width="694" ResizeMode="CanMinimize" Background="#FFC3C3C3" Loaded="
clickme_window_Loaded">
    <Canvas Name="tollesCanvas" HorizontalAlignment="Left" Height="582" Margin="10,10,0,0"
VerticalAlignment="Top" Width="660">
    </Canvas>
</Window>
```

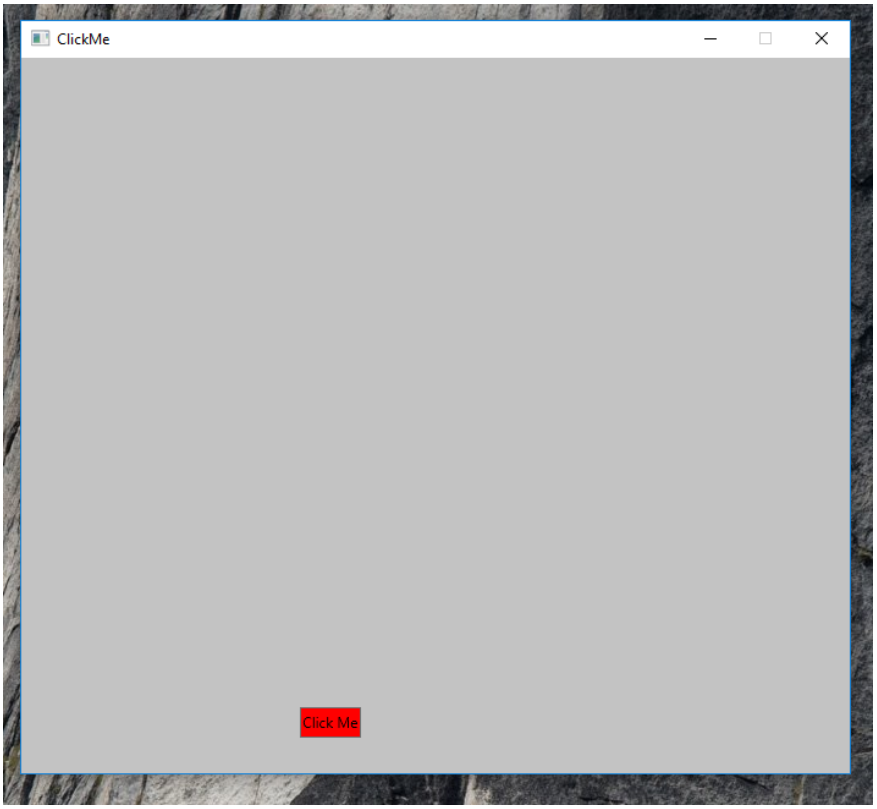
Notiz:

- Ersetzt **"PROJEKTNAME_HIER_EINFUEGEN"** logischerweise mit dem Projektnamen, den Ihr vorher euren Projekt gegeben habt.
- Es ist euch völlig frei überlassen welche Farben ihr wählt oder haben wollt.
- Sowohl dem Fenster, als auch dem Canvas sollten die Ereignishandler "FENSTERNAMEN_Loaded", sowie "CANVASNAMEN_Loaded" hinzugefügt werden. Diese können erzeugt werden, indem man, wenn man das Objekt markiert hat, rechts neben der Textbox zum ändern des Titels auf dem Blitz geht und anschließend auf die Textbox neben "Loaded"

doppelklickt.

- 

Hier ein kleines Bild, unseres Programmes:



Grundlegendes:

Als erstes müssen wir uns natürlich im klaren werden, wie wir dies nun umsetzen. Ich hätte gesagt, wir erstellen einen Timer, der den Button nach jeder Sekunde an einen zufälligen Ort im Fenster verschiebt. Ebenfalls müssen wir einen Zufallsgenerator erstellen, der ein Objekt der Klasse Random entspricht. Dazu müssen wir aber auch noch den Button erstellen. Dies sieht in **Code** wie folgt aus:

Datentypen etc. erstellen

```
                  //Hier der Intervall, wie oft der Timer die Tick Methode ausführen soll, in diesem Fall,
pro Sekunde ein mal
                int interval_int = 1000;
                //Hier erstellen wir den Timer und eine Zeile darunter den Zufallsgenerator
        DispatcherTimer timer = new DispatcherTimer();
        Random rm = new Random();
                //Da wir den Button nicht in WPF direkt erstellen, erstellen wir ihn hier
        Button clickme_button = new Button()
        {
            Content = "Click Me!",
            Height = 25, Width = 50,
            Background = Brushes.Red
        };
```

Danach müssen wir natürlich erstmal festlegen, was denn genau passiert wenn das Programm geöffnet wird und die Windows Form geladen wird. Hier setzt er nämlich den Timer Intervall, startet diesen und setzt den Button an eine zufällige Position. In **Code** sieht das wie folgt aus:

Window Loading

```
 private void clickme_window_Loaded(object sender, RoutedEventArgs e)
        {
                        //In dieser Methode, passiert alles wenn man das Programm startet (z.B. den Timer
starten, den Button an eine Zufällige Position setzen, etc.)
            clickme_button.Click += btnClick;
            timer.Interval = TimeSpan.FromMilliseconds(interval_int);
            timer.Tick += tick;
            timer.Start();
            tollesCanvas.Children.Add(clickme_button);
            set_button_random_pos();
        }
```

Als nächstes müssen wir erstmal programmieren, was denn genau passiert, wenn man auf den Button draufklickt. Ganz oben sagen wir, dass wenn jemand Enter oder Leertaste drückt, soll man den Button nicht drücken können ("Sicherheitsfunktion"). Als nächstes und letztes sagen wir den Programm zunächst bei else, wenn man auf den Button draufklickt, dann soll er den Intervall um 50 verringern. Eins darüber, überprüft er mit if, ob der Intervall kleiner als 51 ist, da es sonst zu einen Error kommen würde. Falls er kleiner ist, weiß das Programm, dass der Spieler gewonnen hat und gibt somit eine Nachricht aus, die der Spieler sieht, in diesem Fall "Glückwunsch, du hast gewonnen!". Danach führt er eine Void aus, die das tut wie sie heißt und zwar, dass Programm neu zu starten (**Der Code dazu kommt im folgenden**).

Button Klick Event

```
private void btnClick(object sender, RoutedEventArgs e)
       {
           if(Keyboard.IsKeyDown(Key.Enter) || Keyboard.IsKeyDown(Key.Space))
           {
               return;
           }

           if (interval_int < 51)
           {
               timer.Stop();
               MessageBox.Show("Glückwunsch, du hast gewonnen!");
               Restart();
           }
           else
           {
               interval_int -= 50;
               set_button_random_pos();
               timer.Interval = TimeSpan.FromMilliseconds(interval_int);
           }
       }
```

Danach, muss dass Programm aber auch ebenfalls wissen, was denn der Timer durchgehend ausführt, dies machen wir in der Methode namens **tick**. Hier führt das Programm nur eine andere Methode namens "set_button_random_pos" aus. Diese bestimmt die 2 Zufallszahlen von den 2 Werten **X** und **Y**. Da es ein bisschen komisch wäre, wenn der Button auf einmal verschwindet, müssen wir die Breite des Buttons anpassen, dies passiert in der dritten Zeile der Methode. Hier der Code:

Timer

```
//Diese Methode ruft unser Timer auf, in unserem Falle jede Sekunde ein mal
private void tick(object sender, EventArgs e)
       {
           set_button_random_pos();
       }

       public void set_button_random_pos()
       {
                       //Hier holen wir uns 2 zufällige Koordinaten für den Button, der nachher zufällig
"herumspringen" soll
           int x_int = rm.Next((int)(tollesCanvas.Width - clickme_button.Width));
           int y_int = rm.Next((int)(tollesCanvas.Height - clickme_button.Height));
           clickme_button.Margin = new Thickness(x_int, y_int, 0, 0);
       }
```

Zu guter letzt, haben wir noch 2 Methoden, eine ist die Methode "Restart" die andere die Methode "Reset". Beide machen im Prinzip das was Ihr Name auch sagt. In der Methode Restart frägt dich das Programm, ob du nach einem Sieg nochmals spielen willst oder nicht, falls du auf "Retry?" klickst, führt er die folgende Methode "Reset" aus, falls du aber auf "Confirmation" klickst, schließt sich das Programm selbst. Nun zu der "Reset" Methode, diese setzt den Timer Intervall auf 1000, den Start-Button wieder an eine zufällige Position, den Intervall Integer ebenfalls auf 1000 und zu guter letzt, startet das Programm wieder den Timer. Hier der Code:

Restart und Reset

```
  public void Restart()
        {
            System.Windows.Forms.DialogResult result = System.Windows.Forms.MessageBox.Show("Retry?",
"Confirmation", System.Windows.Forms.MessageBoxButtons.YesNo);
            if (result == System.Windows.Forms.DialogResult.Yes)
            {
                Reset();
            }
            else if (result == System.Windows.Forms.DialogResult.No)
            {
                Environment.Exit(0);
            }
        }

        private void Reset()
        {
            interval_int = 1000;
            set_button_random_pos();
            timer.Interval = TimeSpan.FromMilliseconds(interval_int);
            timer.Start();
        }
```

Fertig!

Ihr könnt das Programm natürlich auch nach belieben verfeinern und verbessern. Hier ein paar Vorschläge:

- Ein Score-Bestleistungssystem wo Ihr sehen könnt, wer bisher der beste Spieler war
- Verschiedene Längen des Spiels und Schwierigkeitsgrade
- Vielleicht etwas verschönern, sprich z.B. den Button mit einen Bild Designen oder den Hintergrund verschönern