Nicolas Plieninger

TicTacToe

Dokumentation

Contents

[Zielbestimmungen 2](#_Toc136634832)

[Musskriterien 2](#_Toc136634833)

[Wunschkriterien 2](#_Toc136634834)

[Flussdiagramm 3](#_Toc136634835)

[Lösungsansatz 4](#_Toc136634836)

[Vorbereitungen 4](#_Toc136634837)

[Programm 5](#_Toc136634838)

[Testprotokoll 8](#_Toc136634839)

[Test 1: Läuft Programm auf Raspberry Pi 8](#_Toc136634840)

[Test 2: Funktionieren alle Buttons auf dem Touchpanel 8](#_Toc136634841)

[Test 3: Wird ein Unentschieden erkannt 8](#_Toc136634842)

[Test 4: Werden alle Gewinnvariationen erkannt und richtig gewertet 9](#_Toc136634843)

## Zielbestimmungen

### Musskriterien

-Das Hauptziel ist ein vollumfängliches TicTacToe-Spiel, das auf einem Touchscreen gespielt werden kann.

-Eine Spielzuganzeige, welche anzeigt, welcher Spieler gerade an der Reihe ist.

-Einen Resetbutton mit dem man das Spielfeld zurücksetzen kann

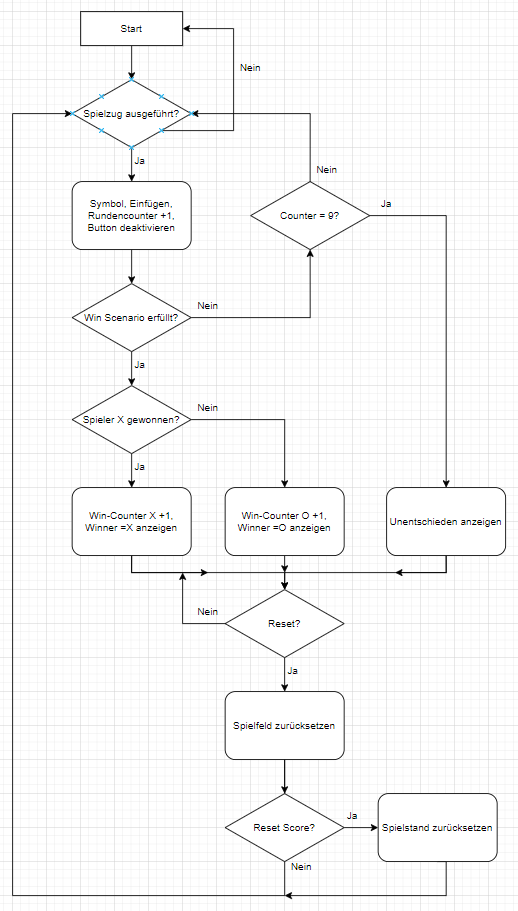
-Einen «Win Counter» der den Spielstand der zwei Spieler anzeigt

### Wunschkriterien

-Eine externe Anbindung (Kommunikation mit einem Drittsystem)

-Eine Gewinnanimation, wenn ein Spieler 10 Punkte erreicht.

# Flussdiagramm



# Lösungsansatz

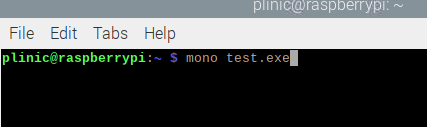
# Vorbereitungen

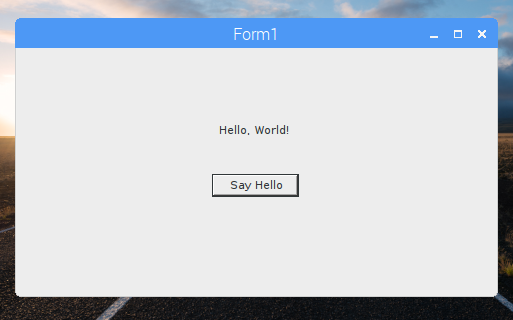
Da wir den Code in C++ schreiben müssen, gilt es als erstes herauszufinden, wie man WinForms auf einem Linux basierten Raspberry Pi laufen lassen kann.

Dabei bin ich auf das Software Framework Mono gestossen.

Um das Ganze zu testen, wurde ein ganz einfaches WinForms Programm erstellt und die .exe Datei auf meinen Raspberry Pi geladen.

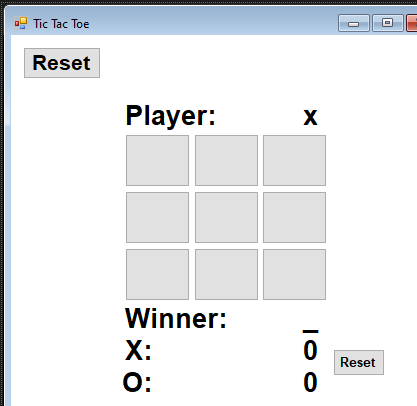
Im Command Terminal kann dann mit folgendem Befehl das Programm über Mono ausgeführt werden.



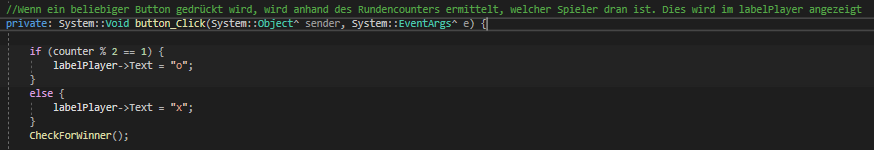


# Programm

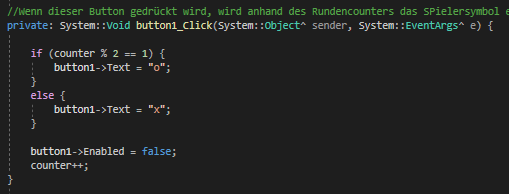
Als erstes wird das GUI aufgebaut:



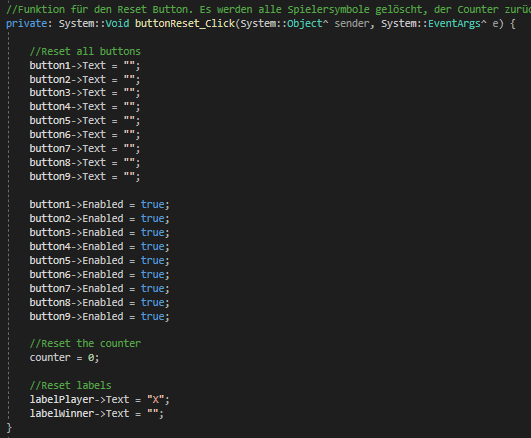
Für die Anzeige welcher Spieler gerade an der Reihe ist, wird ein Rundencounter implementiert. Der Counter beginnt bei 0 und zählt bei jedem Spielzug um 1 hoch. Ist der Counter eine gerade Zahl, wird Spieler «X» angezeigt. Bei den ungeraden Zahlen Spieler «O».



Danach wird für jeden einzelnen Button (hier Beispiel Button1) eine Funktion erstellt. Wird der Knopf betätigt, wird abhängig vom Rundencounter das Symbol des jeweiligen Spielers in den Text eingefügt. Des Weiteren wird der Knopf deaktiviert und der Rundencounter zählt um 1 hoch.

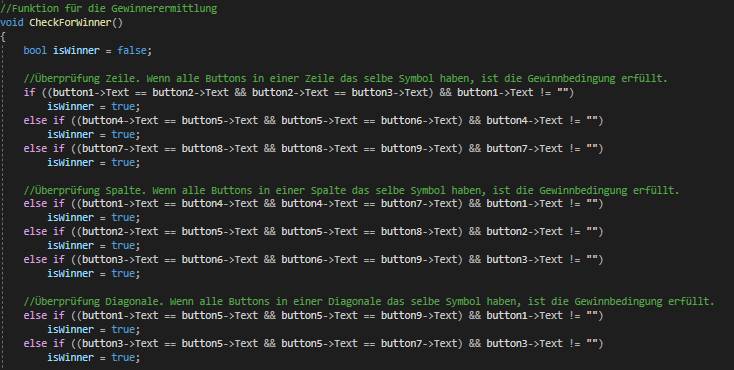


Um das Spielfeld zurückzusetzen, wird der Reset-button implementiert. Dieser macht nichts anderes, als den Rundencounter wieder auf 0 zu setzen, alle Knöpfe wieder zu aktivieren und alle Spielersymbole zu löschen.



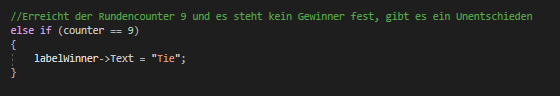
Der Hauptteil des Programmes ist es, die Gewinnszenarien zu ermitteln. Sprich drei gleiche Symbole in einer Zeile, Spalte oder Diagonalen.

Bei jedem Knopfdruck wird im Programm abgefragt, ob es ein Gewinnszenario gibt.



Falls ein solches Szenario auftritt, wird anhand des Rundencounters ermittelt welcher Spieler gewonnen hat.

Falls das Spielfeld gefüllt ist, jedoch kein Gewinnszenario erfüllt wurde, gibt es ein Unentschieden.



Zu guter Letzt gibt es noch die «Win Counter» die bei jedem Sieg einfach dem jeweiligen Spieler einen Punkt gutschreiben. Diese Counters können mit einem eigenen Button zurückgesetzt werden.

# Testprotokoll

## Test 1: Läuft Programm auf Raspberry Pi

In diesem Test wird überprüft, ob das fertige Programm überhaupt auf dem Raspberry Pi gestartet werden kann.

Ergebnis: Nein. Beim Ausführen des Befehls gibt es eine Error Meldung. Grund dafür ist wahrscheinlich, dass nicht alle verwendeten Instructions mit Mono kompatibel sind. Um den Fehler zu beheben, habe ich das gesamte Programm nochmals in C# erstellt. Das zweite Programm liess sich Problemlos starten.

## Test 2: Funktionieren alle Buttons auf dem Touchpanel

Ergebnis: Ja alle neun Spielbuttons und die zwei Reset-Buttons können betätigt werden.

## Test 3: Wird ein Unentschieden erkannt

Ergebnis: Ja ein Unentschieden wird erkannt und kein Counter zählt hoch.

## Test 4: Werden alle Gewinnvariationen erkannt und richtig gewertet

Hier wird überprüft, ob alle 16 möglichen Gewinnvariationen erkannt und richtig gewertet werden.

-Variante 1: Spieler X, Zeile 1

-Variante 2: Spieler X, Zeile 2

-Variante 3: Spieler X, Zeile 3

-Variante 4: Spieler X, Spalte 1

-Variante 5: Spieler X, Spalte 2

-Variante 6: Spieler X, Spalte 3

-Variante 7: Spieler X, Diagonale 1

-Variante 8: Spieler X, Diagonale 2

-Variante 9: Spieler O, Zeile 1

-Variante 10: Spieler O, Zeile 2

-Variante 11: Spieler O, Zeile 3

-Variante 12: Spieler O, Spalte 1

-Variante 13: Spieler O, Spalte 2

-Variante 14: Spieler O, Spalte 3

-Variante 15: Spieler O, Diagonale 1

-Variante 16: Spieler O, Diagonale 2

Ergebnis: Alle Varianten werden erkannt und die Counter zählen richtig hoch.