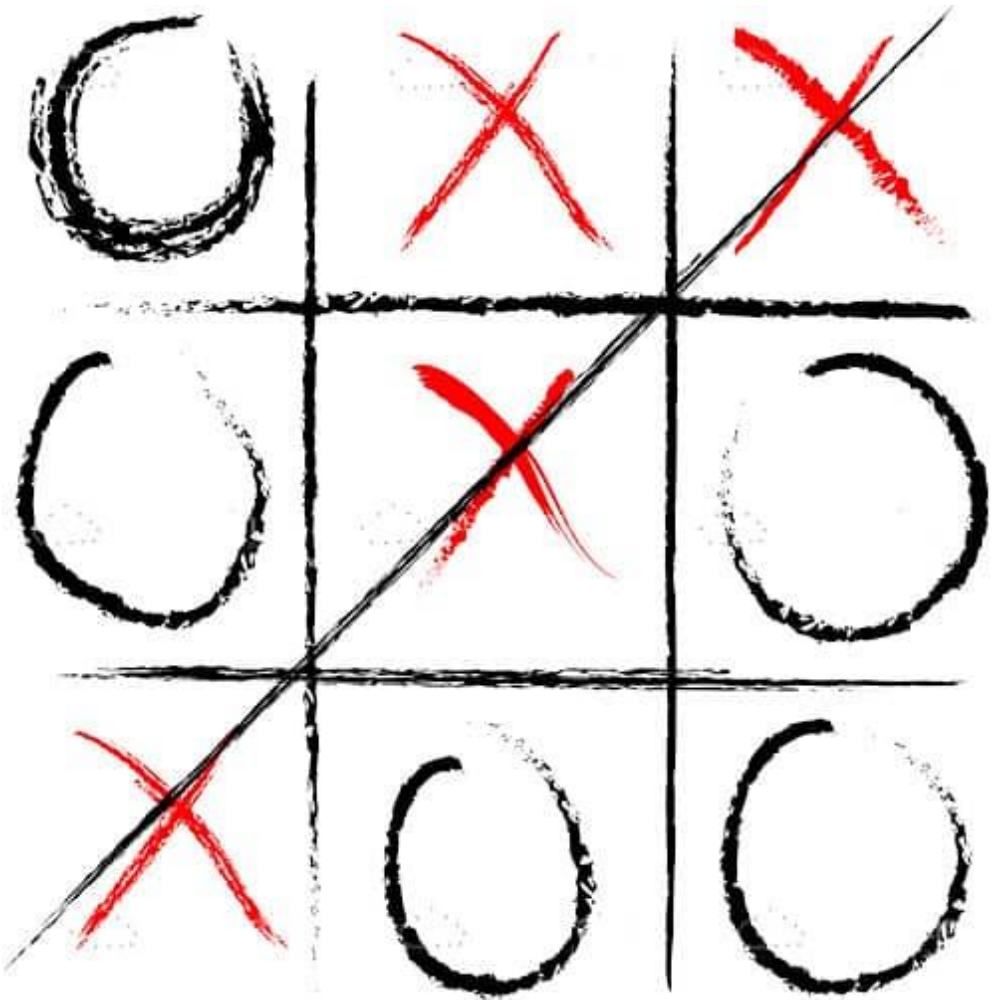
Project Tic Tac Toe



Start datum: 10-02-20 (week 7)
Eind datum:

Door:
Machi, Anchi, Buchi en Juancho (Product Owner)

Voor Informatica Rapport 3
Docent: ing. R. Ellis (ES)

Voorwoord

Dit project is uitgevoerd voor het vak Informatica gegeven door dhr. R. Ellis. Voor dit project moest er een Tic Tac Toe spel ontwikkeld worden.

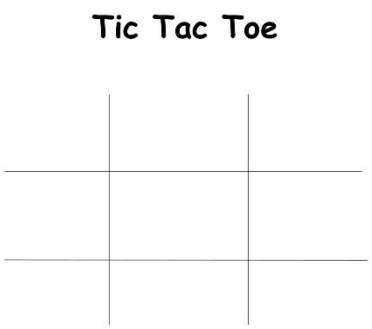
Inhouidsopgave

1 Inleiding

In dit verslag wordt uitgewerkt hoe dit project is uitgevoerd.

1.1 Het spel

Het spel Tic-Tac-Toe wordt normaal door twee spelers op papier gespeeld waarbij er een bord wordt getekend van drie rijen en drie kolommen. Om de beurt plaats een speler of een X (voor speler 1) of een O (voor speler 2) op het bord.



De speler die zij het horizontaal, verticaal of diagonaal drie keer zijn teken heeft kunnen plaatsen heeft gewonnen. Indien er geen lege plekken meer zijn en geen van beide spelers hebben drie van hun tekens op rij kunnen zetten dan is het spel geëindigd in een gelijk spel.

1.2 Middelen

Het programma is ontwikkeld in Visual Studio Community 2019 en moet een Windows Forms desktop applicatie worden.

1.3 Ontwikkel methode

Voor het ontwikkelen van het programma wordt er in een aantal fasen gewerkt die verder in dit verslag zullen worden uitgewerkt. De fasen zijn Planning, Analyse, Ontwerp, Ontwikkeling en testen. Tevens is tijdens het ontwikkelen van de code gebruik gemaakt van de SCRUM methode.

1.4 SCRUM methode

De SCRUM methode is een speciale manier van werken die wordt toegepast bij het ontwikkelen van software. Er wordt gebruik gemaakt van een team van, normaal gesproken, zeven personen met verschillende kwaliteiten en specialiteiten. Eén van de teamleden is de Product Owner. Deze onderhoudt het contact tussen het team en de klant. Een ander teamlid is de Scrum Master, deze zorgt dat alles wat het team nodig heeft aanwezig is. Voor dit project is de Scrum Master de docent.

1.4.1 Sprints

Een scrum bestaat uit één of meerdere Sprints die samen het project moeten voltooien. Er zouden dus meerdere sprints tegelijk kunnen worden uitgevoerd. Normaal duurt een sprint 3 weken en dan moet een onderdeel tot een werkend resultaat gebracht zijn. De onderdelen worden gehaald uit een backlog, dit is een lijst met alle taken die gedaan moeten worden voor het project, en geplaats in een sprint backlog voor uitvoer.

2 Planning

In de tabel hieronder ziet men een overzicht van de planning van het project met de taken die per week (van de jaarkalender) uitgevoerd dienen te worden. Tevens wordt er aangegeven wie aan welk onderdeel gaat werken.

Week	Taken	leden
7	Opdracht bestuderen, Verslagstructuur maken, voorwoord en inleiding schrijven, planning maken.	Alle teamleden
8	Analyse	Alle teamleden
9	Ontwerp	Alle teamleden
10	Ontwikkeling	Zie Scrum bord
14	Testen	Alle teamleden
15	Opleveren	

3 Analyse

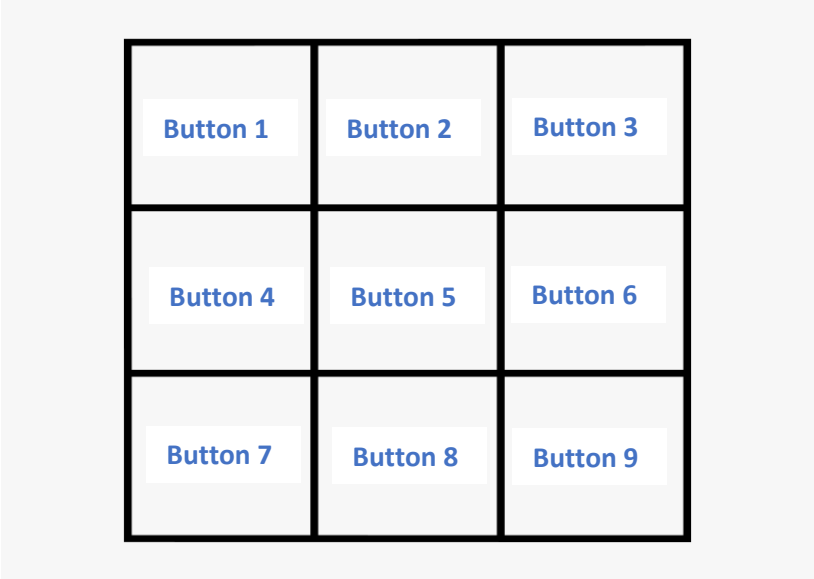
In deze fase wordt gekeken hoe het spel Tic-Tac-Toe gemaakt kan worden in C3 met Windows Forms.

3.1 Interface en scherm lay-out

Programma's die gemaakt zijn met Windows Forms kunnen gebruik maken van de verschillende grafische Windows objecten, zoals buttons, labels, menus, lists, radio buttons etc. Voor dit programma zal er gebruik gemaakt worden van labels en buttons.

Labels (tekstvakken) zijn vakken waar een tekst in geplaats kan worden die de gebruiker kan zien. Buttons zijn knoppen die de gebruiker van het programma kan aan klikken waarna het programma een actie kan ondernemen.

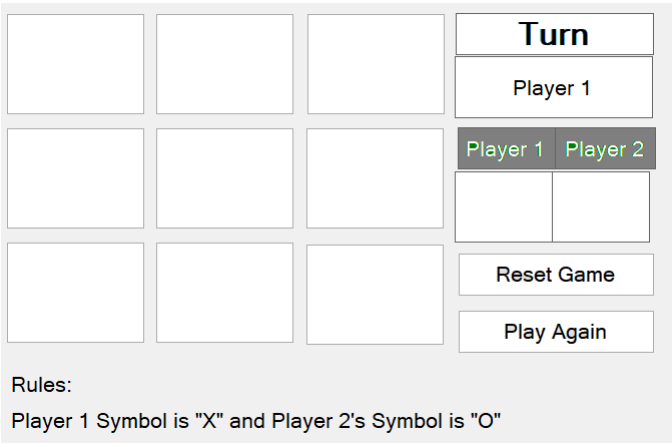
Het speelscherm zal bestaan uit 9 buttons die kunnen worden aangeklikt door de gebruiker. Zie de afbeelding hier onder.



Elke knop kan drie toestanden hebben:

1. Leeg
2. Een 'X'
3. Een 'O'

Naast het bord komen er tekstvakken om aan te geven welke speler aan de beurt is end hoeveel keer elke speler gewonnen heeft. Tevens zullen er knoppen zijn om het spel te her-starten of om opnieuw te spelen. Als laatste zal er nog een tekst vak zijn om de regels aan te geven. Schematisch zal de lay-out van het scherm er dus uitgaan zien zoals in de afbeelding hiernaast.



3.2 Knoppen status

Daar C# een object georiënteerde en event driven programmeertaal is kun je een actie bepalen al een object gemanipuleerd wordt. Dus wanneer de gebruiker op een knop klikt activeert (trigger) dit een om een specifieke functie aan te roepen. In deze functie moet gecontroleerd worde welke status de knop heeft. Afhankelijk van de status moet er dan gecontroleerd worden of het spel gewonnen is door die speler of dat het een gelijk spel is of dat de andere speler de beurt krijgt om een zet te doen.
Als de status van de knop "Leeg" is dan krijgt het de waarde van de speler die aan zet is. Als er al door een speler op die knop geklikt heeft dan kan er een foutmelding gegeven worden, in de form van een geluid, of er gebeurd niets.

4 Ontwerp

In deze fase worden de verschillende onderdelen verder uitgewerkt. Er wordt bepaald welke variabelen gebruikt worden, wat de namen zijn van de objecten die gebruikt gaan worden en welke functies noodzakelijk zullen zijn.

Daar het voornamelijk om events gaat zijn er verder geen parameters bij de event extra dan die Visual Studio zelf aanmaakt Ook heeft geen van de functies een return waarde.

4.1 Variabelen

Voor het project zullen de volgende (groepen) variabelen gebruikt worden

De variabele **turn** van het type integer. Deze heeft als begin waarde 1. Het zal gebruikt worden om de beurten van de twee spelers bij te houden. Speler 1 zal de oneven beurten hebben en speler 2 de even beurten.

De groep van variabele **click1** t/m click **click9** van het type integer. Deze zullen allemaal een beginwaarde hebben van 0 om aan te geven dat die knop nog niet is aangeklikt.

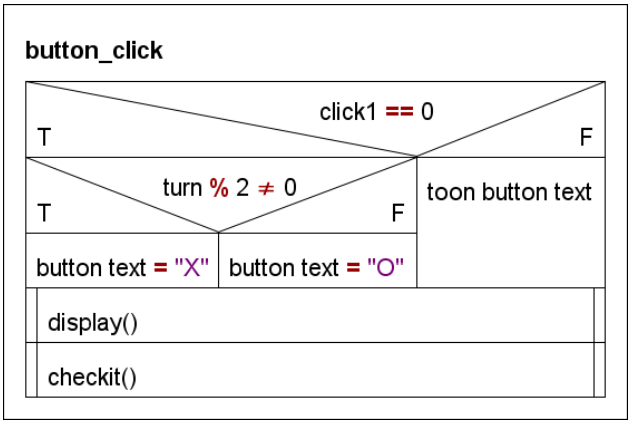
De derde groep variabelen bestaat uit **player1** en **player2**. Beide van het type integer met begin waarde 0.

4.2 Functies

Voor de werking van het programma zijn er een aantal functies noodzakelijk. Sommige zijn Event functies, bijvoorbeeld het klikken door de gebruiker op een knop en sommige zijn hulp functies.

4.2.1 Knop event functies

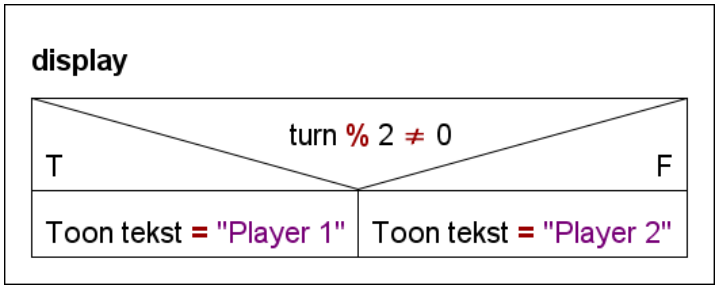
In een Windows Forms programma wordt er automatisch een Event functie aangemaakt door Visual Studio. Voor elk van de negen knoppen van het spelbord is er dus één event. Wat deze events moeten doen is in principe identiek.
Dus voor het **button1_Click** event wordt er een functie in Visual Studio aan gemaakt. Deze moet dan bepalen of het de eerste keer is dat deze knop wordt aangeklikt. De variabele click1 heeft dan dus de waarde 0. Als dit het geval is dan moet de functie bepalen of het speler 1 of speler 2 was die de knop had ingedrukt. Dit wordt gedaan deer te kijken naar de rest waarde *(modulo %)* van de variabele **turn**. Als de rest waarde bij deling door 2 nul is dan betekent het dat speler 2 op de knop had gedrukt. Anders was het speler 1. Hierna worden de variabelen turn en click1 elk met 1 verhoogd.



Daarna worden twee functies aangeroepen. De functie **display()** om de zet waardes te veranderen en de functie **checkit()** om te controleren of de speler gewonnen heeft, of dat een gelijkspel is of dat speler twee aan de beurt is om een zet te doen. Zie die afbeelding van de PSD.

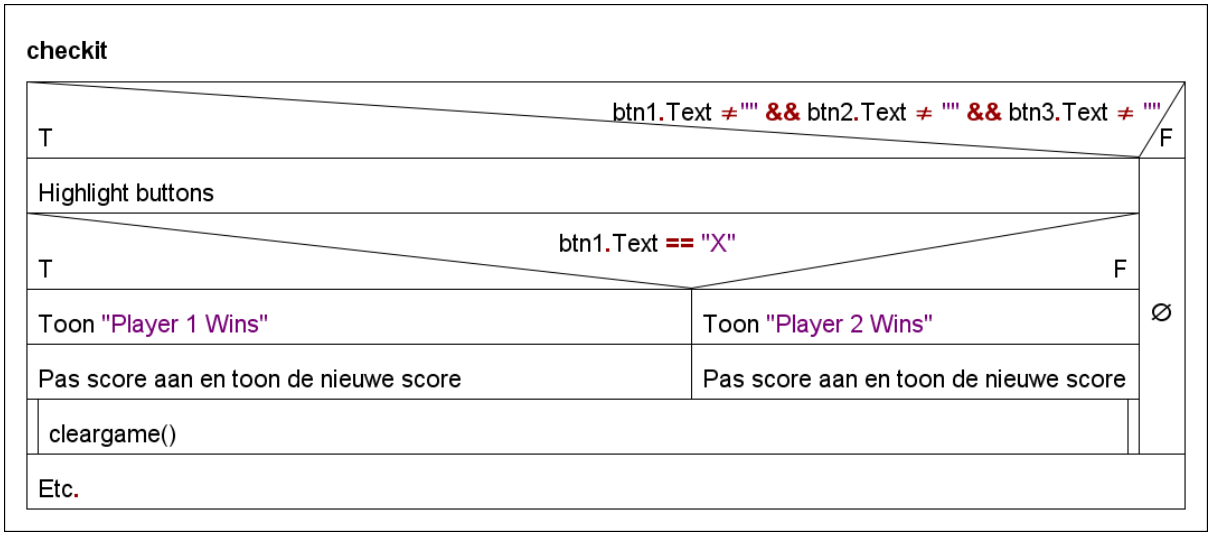
4.2.2 De functie display()

De functie display() past de tekst op in het tekst vak aan om aan te geven welke speler aan de beurt is. Zie de afbeelding van het PSD hier naast.



4.2.3 De functie checkit()

In deze functie wordt er gecontroleerd of de speler die net een zet gedaan heeft gewonnen heeft. Hierbij worden alle mogelijke win situaties bekeken. Als dit het geval is wordt de score van de betreffende speler aan gepast in de variabele en getoond op het scherm. Een deel van de PSD ziet er uit zo als in de afbeelding.



4.2.4 De functie cleargame()

Deze functie zet alles terug naar de begin waarden behalve se score van de spelers.

4.2.5 Het klik event van de reset knop

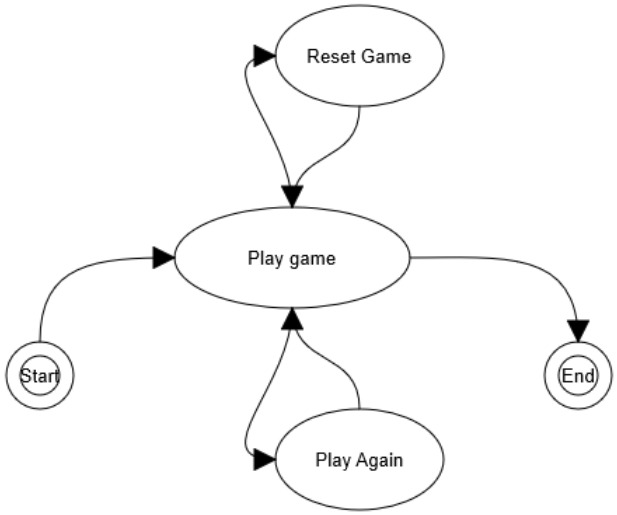
Deze zet alle programma waarden terug naar de begin toestand van het programma. Ook de speler scores.

4.2.6 Het klik event van de playagain knop

Deze start een nieuw spel met de zelfde spelers. De speler scores blijven behouden.

4.3 Programma toestanden

Voor het aangeven van de toestanden waar in het programma zich kan bevinden wordt gebruik gemaakt van een **state diagram**. De onderstaande afbeelding geeft het state diagram voor dit programma weer.



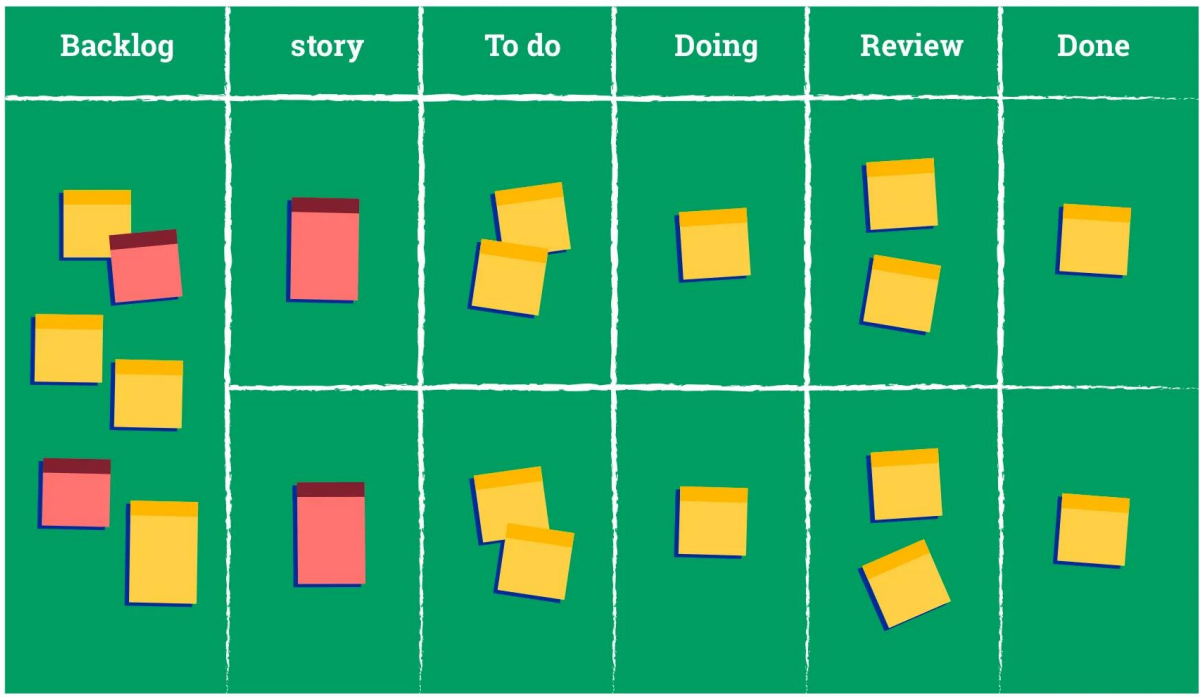
4.4 Taak verdeling

In de onderstaande tabel wordt aangegeven wie welk onderdeel al taak heeft om uit te werken.

Taken	Teamlid
Interface, playagain event	Machi
Cleargame(), reset event	Anchi
Display(), checkit()	Buchi
Button_click events	Juancho

5 Ontwikkeling

Tijdens het ontwikkelen is gebruik gemaakt van de Scrum methode. Deze methode gebruikt een Scrum Bord waarop aangegeven wordt welke taken gedaan moeten worden, welke men mee bezig is en welke klaar zijn. Een soort van Todo list. Op een bord worden met sticky notes de taken op het bord geplakt. De notes worden verschoven van de ene kolom naar de de andere om aan te geven waar men mee bezig of wat al klaar is. Aan het eind van de sprint periode moeten alle taken klaar zijn. De onderstaande afbeelding geeft een idee van hoe dat er uit zou kunnen zien.



Digitaal zag het begin van het Scrumbord er zo uit:

To do	Doing	Done
Interface		
Button_Click events		
display()		
checkit()		
Reset event		
Play again event		
Cleargame()		

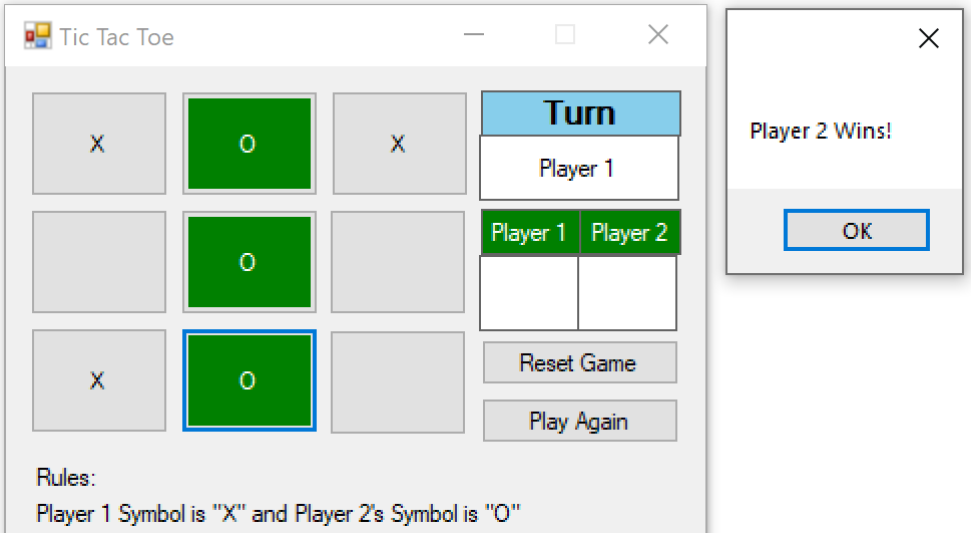
6 Testen

In de test fase zijn alle eisen aan het programma getest of ze werken. Dit wordt ook wel de acceptatie test genoemd.
Er is gecontroleerd dat de score goed wordt bij gehouden. De waarbij een speler wint of er is een gelijkspel zijn getest. Het opnieuw spelen en het her-starten is getest. Alles functioneert naar behoren.

Tijdens het ontwikkelen en testen zijn er meerdere keren kleinere fouten verholpen. De meeste waren syntax fouten. Dit wil zeggen dat instructies niet goed waren geschreven of dat er komma's of haakjes ontbraken. Deze konden snel verholpen worden daar Visual Studio daar goede indicaties voor geeft.

Lastiger waren fouten waar de variabele waarden niet klopten waardoor het programma niet naar behoren reageerde. Door gebruik te maken van het debugger systeem van Visual Studio konden ook deze redelijk vlot worden opgelost.

Hieronder worden er afbeeldingen getoond van het werkende programma.



Bijlagen

Bijlage A: Logboek

Week	Teamlid
1	Juanchi structuur verslag
	Machi voorwoord en inleiding
	Buchi en Anchi lay-out verschal
2	Buchi en Anchi lay-out scherm
	Juanchi aanklik event van 9 buttons
	Machi reser en play again functies
	Etc.

Bijlage B Broncode per functie

Initialisatie variabelen:

```
int turn = 1;
int click1 = 0, click2 = 0, click3 = 0, click4 = 0, click5 = 0, click6 = 0, click7 =
0, click8 = 0, click9 = 0;
int player1 = 0, player2 = 0;
```

Button_Click events:
Hier wordt de code van één event getoond. De andere zijn identiek, alleen de button nummers zullen anders zijn.

```
private void button1_Click(object sender, EventArgs e)
 {
     if (click1 == 0)
     {
         if (turn % 2 != 0)
         {
             button1.Text = "X";
         }
         else
         {
             button1.Text = "O";
         }
         turn++;
         click1++;
     }
     else
     {
         button1.Text = button1.Text;
     }
     display();
     checkit();
 }
```

Display functie:

```
public void display()
 {
     if (turn % 2 != 0)
     {
         displayturn.Text = "Player 1";
     }
     else
     {
         displayturn.Text = "Player 2";
     }
 }
```

Cleargame functie:

```
public void cleargame()
{
    displayturn.Text = "Player 1";
    turn = 1;
    click1 = 0; click2 = 0; click3 = 0; click4 = 0; click5 = 0; click6 = 0;
    click7 = 0; click8 = 0; click9 = 0;
    button1.Text = "";
    button2.Text = "";
    button3.Text = "";
    button4.Text = "";
    button5.Text = "";
    button6.Text = "";
    button7.Text = "";
    button8.Text = "";
    button9.Text = "";
    button1.BackColor = Color.Empty;
    button1.ForeColor = Color.Black;
    button1.UseVisualStyleBackColor = true;
    button2.BackColor = Color.Empty;
    button2.ForeColor = Color.Black;
    button2.UseVisualStyleBackColor = true;
    button3.BackColor = Color.Empty;
    button3.ForeColor = Color.Black;
    button3.UseVisualStyleBackColor = true;
    button4.BackColor = Color.Empty;
    button4.ForeColor = Color.Black;
    button4.UseVisualStyleBackColor = true;
    button5.BackColor = Color.Empty;
    button5.ForeColor = Color.Black;
    button5.UseVisualStyleBackColor = true;
    button6.BackColor = Color.Empty;
    button6.ForeColor = Color.Black;
    button6.UseVisualStyleBackColor = true;
    button7.BackColor = Color.Empty;
    button7.ForeColor = Color.Black;
    button7.UseVisualStyleBackColor = true;
    button8.BackColor = Color.Empty;
    button8.ForeColor = Color.Black;
    button8.UseVisualStyleBackColor = true;
    button9.BackColor = Color.Empty;
    button9.ForeColor = Color.Black;
    button9.UseVisualStyleBackColor = true;
}
```

CheckIt functie

```
public void checkit()
{
    if (button1.Text != "" && button2.Text != "" && button3.Text != "")
    {
        if (button1.Text == button2.Text && button1.Text == button3.Text)
        {
            button1.BackColor = Color.Green;
            button1.ForeColor = Color.White;
            button2.BackColor = Color.Green;
            button2.ForeColor = Color.White;
            button3.BackColor = Color.Green;
            button3.ForeColor = Color.White;
            if (button1.Text == "X")
```

```
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();
        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
            player2++;
            player2score.Text = player2.ToString();
        }
        cleargame();

    }
}
if (button4.Text != "" && button5.Text != "" && button6.Text != "")
{
    if (button4.Text == button5.Text && button4.Text == button6.Text)
    {
        button4.BackColor = Color.Green;
        button4.ForeColor = Color.White;
        button5.BackColor = Color.Green;
        button5.ForeColor = Color.White;
        button6.BackColor = Color.Green;
        button6.ForeColor = Color.White;
        if (button4.Text == "X")
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();

        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
            player2++;
            player2score.Text = player2.ToString();

        }
        cleargame();
    }
}
if (button7.Text != "" && button8.Text != "" && button9.Text != "")
{
    if (button7.Text == button8.Text && button7.Text == button9.Text)
    {
        button7.BackColor = Color.Green;
        button7.ForeColor = Color.White;
        button8.BackColor = Color.Green;
        button8.ForeColor = Color.White;
        button9.BackColor = Color.Green;
        button9.ForeColor = Color.White;
        if (button7.Text == "X")
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();

        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
```

```
            player2++;
            player2score.Text = player2.ToString();

        }
        cleargame();
    }
}
if (button1.Text != "" && button4.Text != "" && button7.Text != "")
{
    if (button1.Text == button4.Text && button1.Text == button7.Text)
    {
        button1.BackColor = Color.Green;
        button1.ForeColor = Color.White;
        button4.BackColor = Color.Green;
        button4.ForeColor = Color.White;
        button7.BackColor = Color.Green;
        button7.ForeColor = Color.White;
        if (button1.Text == "X")
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();

        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
            player2++;
            player2score.Text = player2.ToString();

        }
        cleargame();
    }
}
if (button2.Text != "" && button5.Text != "" && button8.Text != "")
{
    if (button2.Text == button5.Text && button2.Text == button8.Text)
    {
        button2.BackColor = Color.Green;
        button2.ForeColor = Color.White;
        button5.BackColor = Color.Green;
        button5.ForeColor = Color.White;
        button8.BackColor = Color.Green;
        button8.ForeColor = Color.White;
        if (button2.Text == "X")
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();

        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
            player2++;
            player2score.Text = player2.ToString();

        }
        cleargame();
    }
}
if (button3.Text != "" && button6.Text != "" && button9.Text != "")
```

```
{
    if (button3.Text == button6.Text && button3.Text == button9.Text)
    {
        button3.BackColor = Color.Green;
        button3.ForeColor = Color.White;
        button6.BackColor = Color.Green;
        button6.ForeColor = Color.White;
        button9.BackColor = Color.Green;
        button9.ForeColor = Color.White;
        if (button3.Text == "X")
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();

        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
            player2++;
            player2score.Text = player2.ToString();

        }
        cleargame();
    }
}
if (button1.Text != "" && button5.Text != "" && button9.Text != "")
{
    if (button1.Text == button5.Text && button1.Text == button9.Text)
    {
        button1.BackColor = Color.Green;
        button1.ForeColor = Color.White;
        button5.BackColor = Color.Green;
        button5.ForeColor = Color.White;
        button9.BackColor = Color.Green;
        button9.ForeColor = Color.White;
        if (button1.Text == "X")
        {
            MessageBox.Show("Player 1 Wins!");
            player1++;
            player1score.Text = player1.ToString();

        }
        else
        {
            MessageBox.Show("Player 2 Wins!");
            player2++;
            player2score.Text = player2.ToString();

        }
        cleargame();
    }
}
if (button3.Text != "" && button5.Text != "" && button7.Text != "")
{
    if (button3.Text == button5.Text && button3.Text == button7.Text)
    {
        button3.BackColor = Color.Green;
        button3.ForeColor = Color.White;
        button5.BackColor = Color.Green;
        button5.ForeColor = Color.White;
        button7.BackColor = Color.Green;
```

```
                button7.ForeColor = Color.White;
                if (button3.Text == "X")
                {
                    MessageBox.Show("Player 1 Wins!");
                    player1++;
                    player1score.Text = player1.ToString();

                }
                else
                {
                    MessageBox.Show("Player 2 Wins!");
                    player2++;
                    player2score.Text = player2.ToString();

                }
                cleargame();
            }
        }
    }
```

Reset event

```
    private void reset_Click(object sender, EventArgs e)
     {
         player1score.Text = "";
         player2score.Text = "";
         player1 = 0;
         player2 = 0;
         cleargame();
     }
```

PlayAgain event

```
    private void playagain_Click(object sender, EventArgs e)
     {
         cleargame();
     }
```