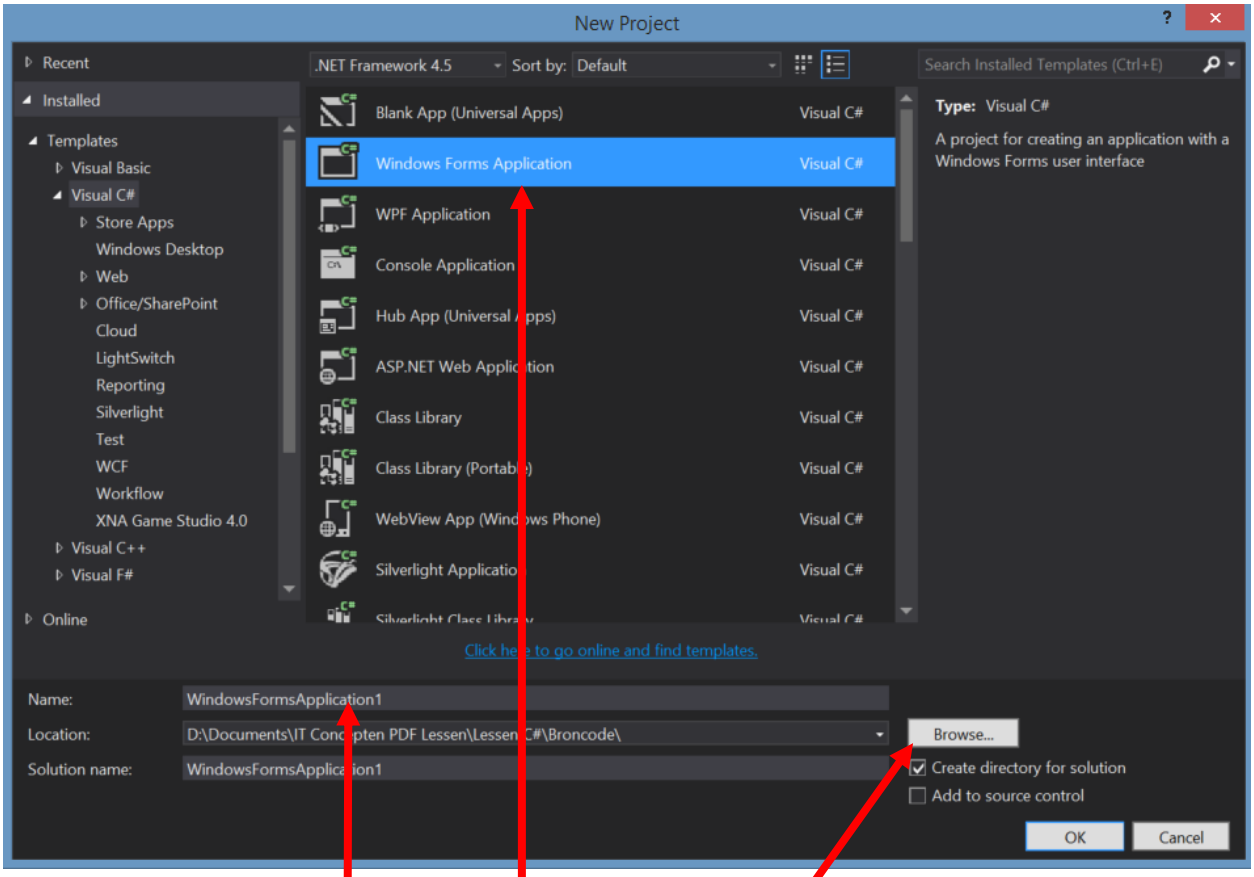
Windows Forms

Windows Forms

In de lessen die tot nu toe geweest zijn is er gewerkt met het Console venster om de basis begrippen van OOP en C# duidelijk te maken. In deze les wordt er gekeken naar het maken van een programma met de Windows Forms.

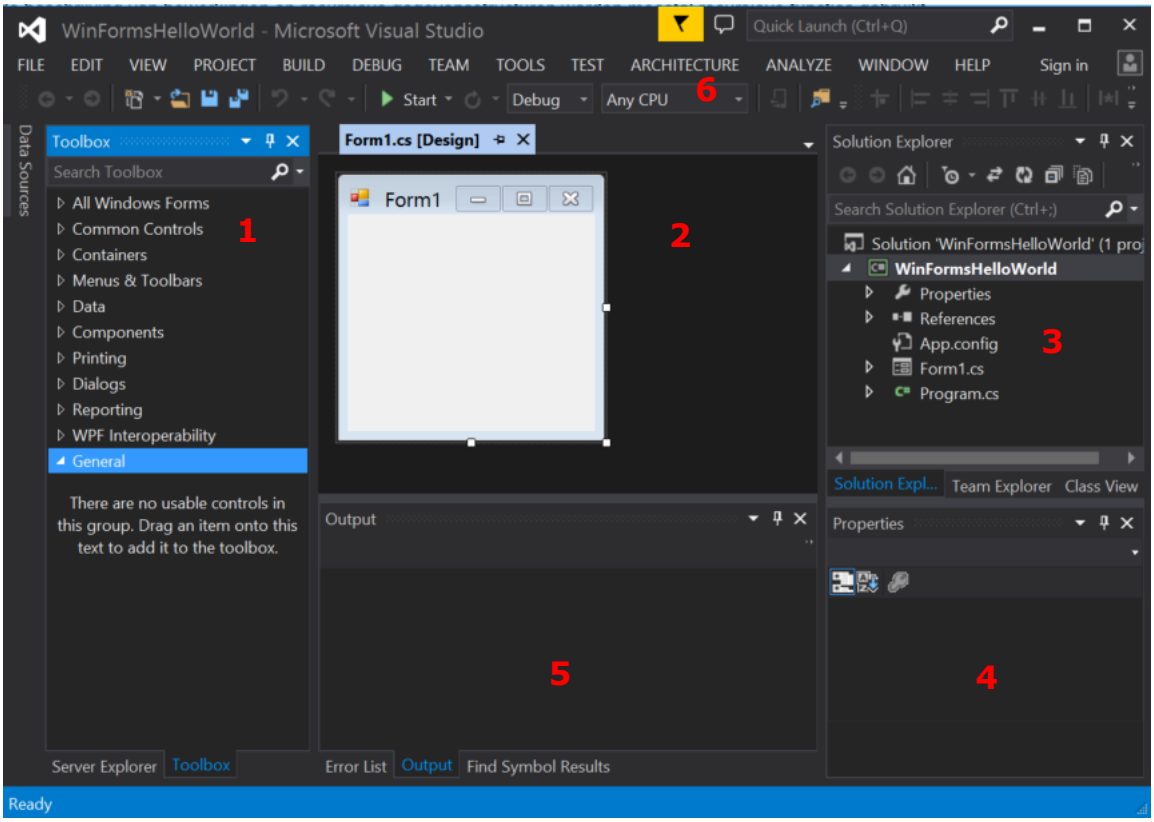
Het “Hello world!” programma met Windows Forms.



1. Start Visual Studio
2. Begin een nieuw project
3. Kies Windows Forms Application
4. Geef als applicatie naam WinFormHelloWorld op
5. Kies de map waar je het wilt opslaan op.
6. Klik op de OK knop.

Windows Forms

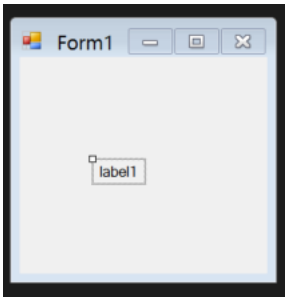
Afhankelijk van je instellingen in Visual Studio krijg je het volgende scherm te zien:



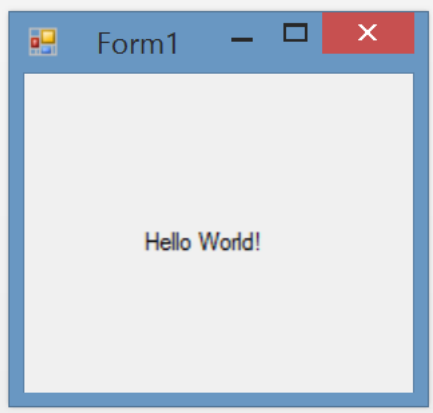
De onderdelen zijn

1. Toolbox
2. Tabbed venster voor broncode
3. Solution Explorer
4. Properties
5. Uitvoer venster
6. IDE menu's en knoppenbalk

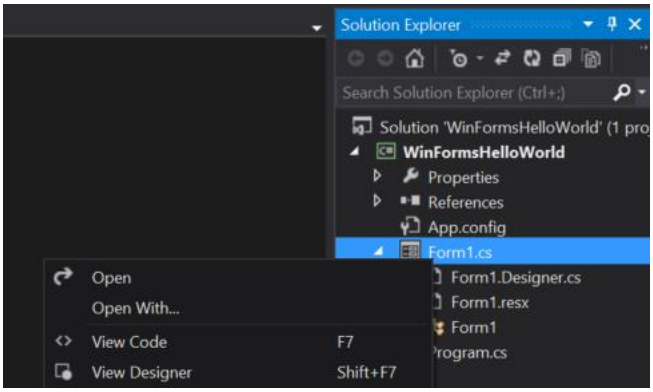
7. In de Toolbox breidt de optie Common Controls uit door op het driehoekje ervoor te klikken.
8. Selecteer een label en sleep het op het venster, "Form1".



9. Met het label component geselecteerd, verander de tekst label1 (deze vind je bij het veld Tekst) in “Hello World!”.
10. In het menu Debug, kies de opties **Start Without Debugging** of druk de toetsen combinaties [CTRL + F5] in.
11. Het programma wordt uitgevoerd. Die ziet er als volgt uit:



In de Solution Explorer, klik rechts op Form1.cs en kies de optie **View Code** of druk op de functie toets [F7].



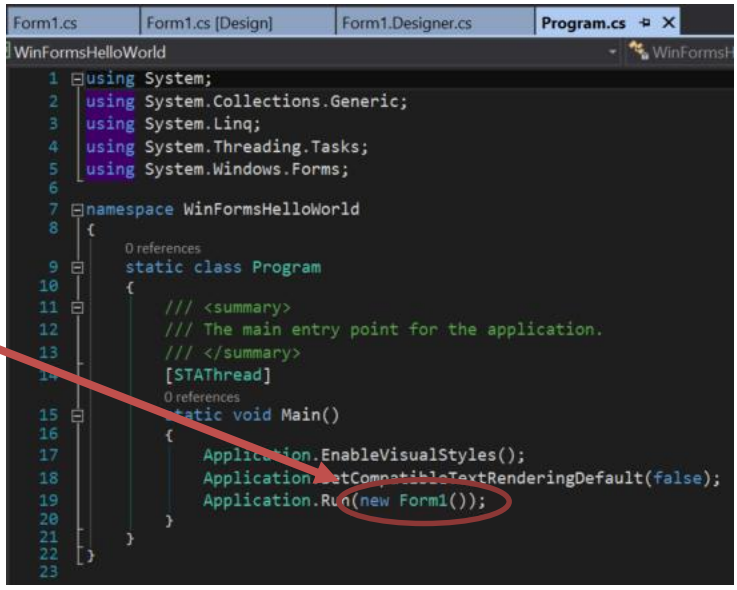
Doe hetzelfde voor Program.cs. Je krijgt zo de broncode te zien van de twee bestanden.

Er is geen enkele broncode door jezelf ingevoerd. Alle code is gegenereerd door Visual Studio.

Windows Forms

Program.cs

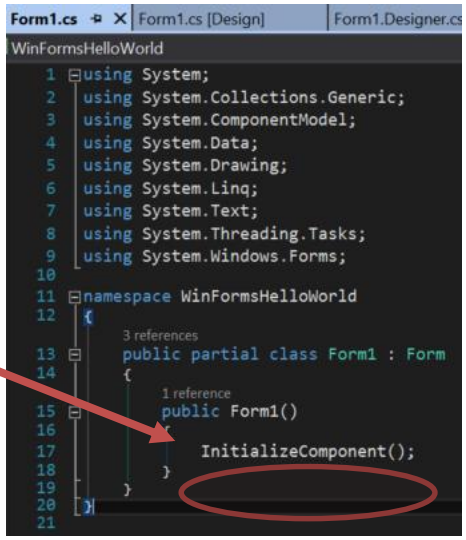
De code in het bestand Program.cs zorgt er voor dat het programma wordt opgestart en dat de control Form1 wordt aangeroepen.



Form1.cs:

In het bestand Forms.cs staat de code om het venster op het beeldscherm te kunnen laten zien met alle componenten die er op zijn aangebracht.

Het aanmaken van de componenten gebeurt in de methode InitializeComponent().



InitializeComponent()

In de Solution Explorer search veld type in InitializeComponent en dubbel klik op de Method om de code te zien.

```
private void InitializeComponent()
{
    this.label1 = new System.Windows.Forms.Label();
    this.SuspendLayout();
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(86, 119);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(94, 20);
    this.label1.TabIndex = 0;
    this.label1.Text = "Hello World!";
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(278, 244);
    this.Controls.Add(this.label1);
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);
    this.PerformLayout();

}
```

Er is daar code gegenereerd om de form en de label aan te maken. Je kunt de eigenschappen zien zoals de coördinaten en de grootte van het component en de naam ervan.

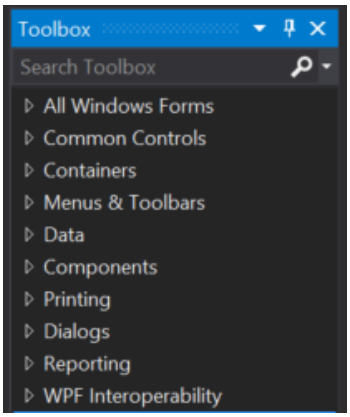
Event driven

Windows Forms programmeren is wat **Event Driven** genoemd wordt. Dit wil zeggen het programma reageert op dingen die gebeuren met het programma door de gebruiker ervan.

Stel dat er op het formulier een knop zou zijn en de gebruiker drukt er op. De gebruiker verwacht dat het programma dan iets gaat doen. Het indrukken van Event ofwel een gebeurtenis waar het programma op moet reageren.

ToolBox

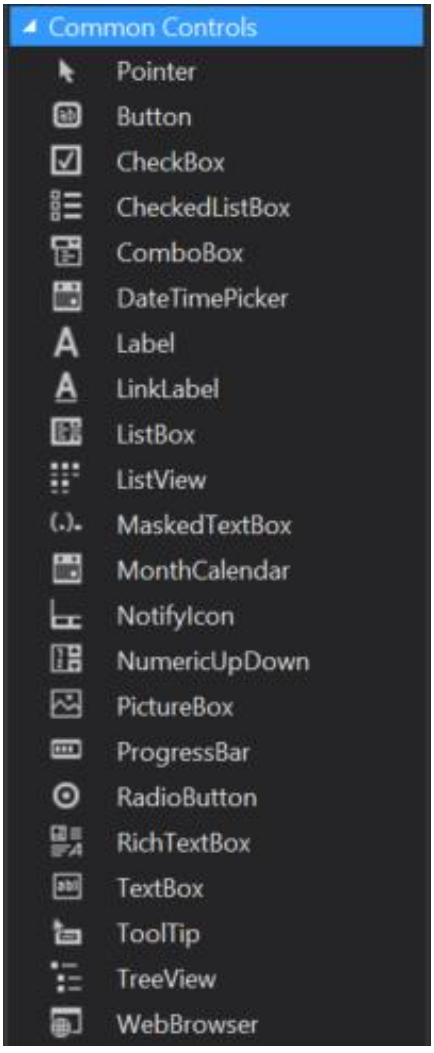
In de ToolBox worden de componenten verdeeld in verschillende categorieën zoals in de afbeelding hiernaast te zien is.



Door op de driehoekjes voor elke categorie te drukken krijg je de verschillende onderdelen behorende tot die categorie te zien.

De categorieën zijn verdeeld aan de hand van de functie van de onderdelen.

Er zijn veel te veel componenten om in deze basisles allemaal te behandelen. Er zal alleen gekeken worden naar een paar veel voorkomende componenten.



Eén hebben we al gezien dat is de **label**. Deze wordt gebruikt om tekst op het formulier te kunnen plaatsen.

Button (knop): Deze komt voor in elk Windows programma. Als er op een knop gedrukt wordt moet er iets gebeuren.

Tekst Box (Tekstveld): Deze laat een gebruiker een tekst invoeren zoals een naam, adres, telefoonnummer etc.

List Box: Deze is handig om de gebruiker een lijst van dingen te laten zien.

Combo Box: De Combo Box Control is een combinatie van een List Box en een Text Box Control. De gebruiker kan een nieuw ding toevoegen via de tekst box of iets selecteren uit de lijst.

Radio buttons: Radio buttons geven de gebruiker de mogelijkheid om een keuze te maken uit ***één*** van de vele opties.

Checkbox: Met Checkboxen kan de gebruiker nul, één of meerdere opties selecteren.

.Net Framework Namespaces

De .NET Framework bibliotheek is enorm groot. Er zijn een groot aantal van namespaces die bruikbare componenten bevatten voor je programma's.

In de tabel hieronder zie je een beschrijving van een aantal veel gebruikte .NET namespaces.

Namespace	**Omschrijving**
System	Definieert exceptions, variable data types, mathematical operations, en andere taal centrale elementn.
System.Collections	Definieert **Lists**, **Maps**, en andere data structuren.
System.Data	Bevat klassen die je toegang geven tot databases
System.Drawing	Laat je kunnen tekenen op de Forms.
System.IO	Ondersteunt lezen en schrijven van data naar bestanden
System.Media	Bevat methods om geluid en muziek te kunnen spelen.
System.NET	Laat je Internet berichten kunnen versturen en ontvangen.
System.Text	Bevat verschillende manieren om tekst te kunnen manipuleren
System.Timers	Definieert het Timer object, welke bruikbaar is bij grafisch porgramma's om dingen te laten gebeuren in bepaalde tijdsintervallen.
System.Web	Definieert objecten die web browser methodes kunnen uitvoeren
System.Windows	De poort in de Windows Presentation Framework (WPF), dit is wat je gebruikt om windows-based grafische programma's te maken.
System.Windows.Forms	Specifieke elementen gerelateerd aan het op dialog-based forms en controls te maken.

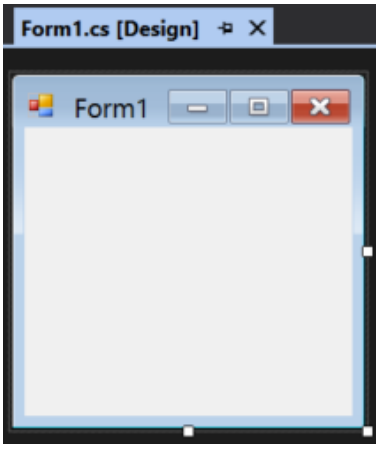
Tic-Tac-Toe

Iedereen kent het spel Tic-Tac-Toe. Daarom wordt dit spel gebruikt als voorbeeld voor een Windows Forms programma. Voor de gene die het ECHT niet kennen hier een korte beschrijving.

Het spel wordt normaliter gespeeld op een rooster van drie bij drie op paper getekend. Twee spelers zetten om de beurt een X of een O in één van de hokjes. De speler die het als eerste lukt om drie van dezelfde tekens in horizontale, diagonale of verticale rij te plaatsen, wint het spel.

Laten we beginnen

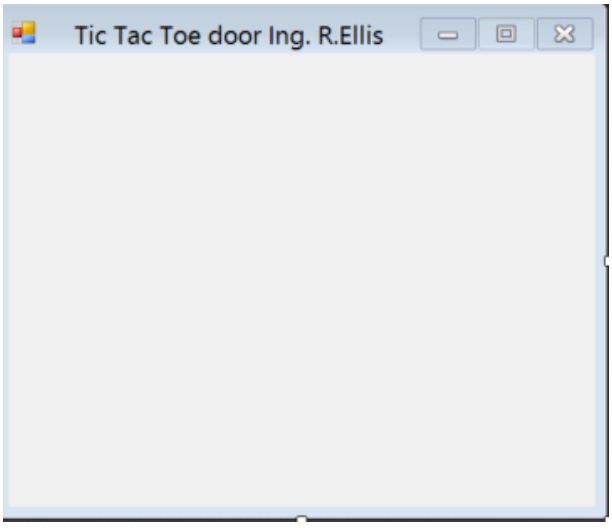
Start Visual Studio en maak een nieuw Windows Applicatie. Geeft het een naam en kies het pad waar het dient te worden opgeslagen.
Een blanco form wordt getoond in VS.



Ga naar het Properties paneel. Wijzig de volgende eigenschappen:

Text: Tic Tac Toe
FormBorderStyle: FixedSingle
Size: 546,462
MaximizeBox: False

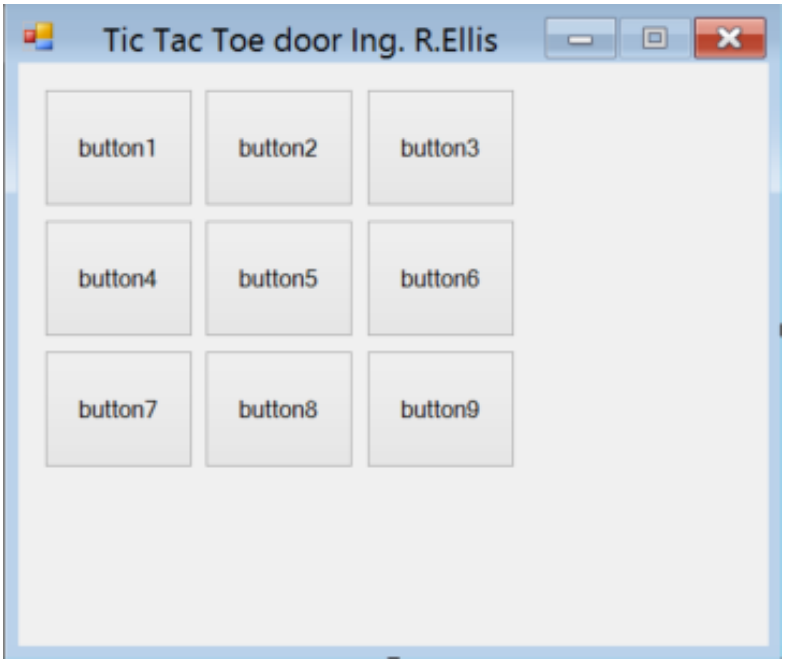
Het venster ziet er dan als volgt uit:



Voeg nu 9 buttons toe met behulp van de tools in de ToolBox. Het venster moet er dan als volgt uit zien.

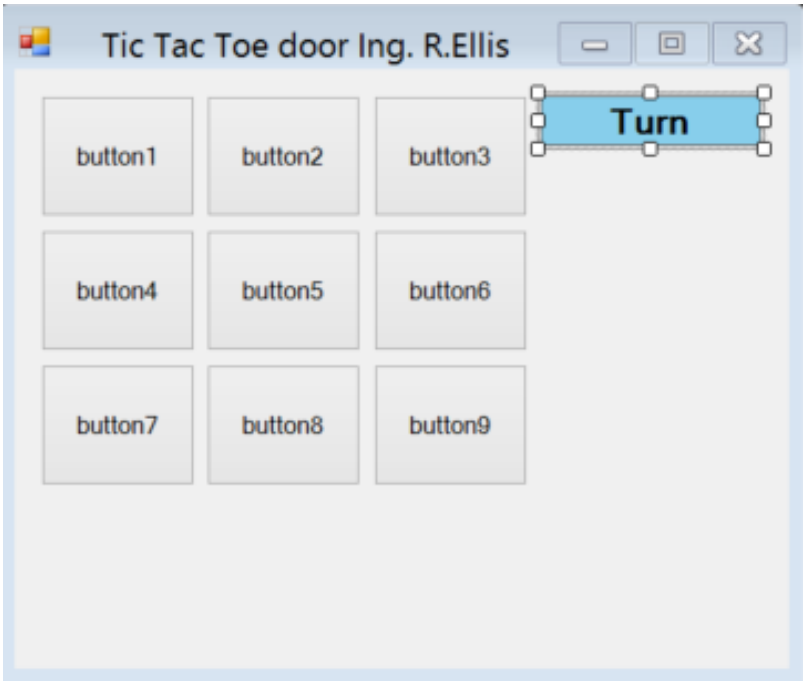
Maak de eerste knop en wijzig de eigenschappen. Daarna kopieer en plak de andere knoppen en wijzig de eigenschappen naar behoren. De eigenschappen voor elk van de negen knoppen:

Text: {leeg maken}
Location: (btn1=18, 18), (btn2=130, 18), (btn3=243, 18),
(btn4=18, 109), (btn5=130, 109), (btn6=243, 109),
(btn7=18, 200), (btn8=130, 200), (btn9=243, 200)
Margin: 4,5,4,5
Size: 104, 82



Denk eraan verwijder de tekst button1…9 in de eigenschap Text.

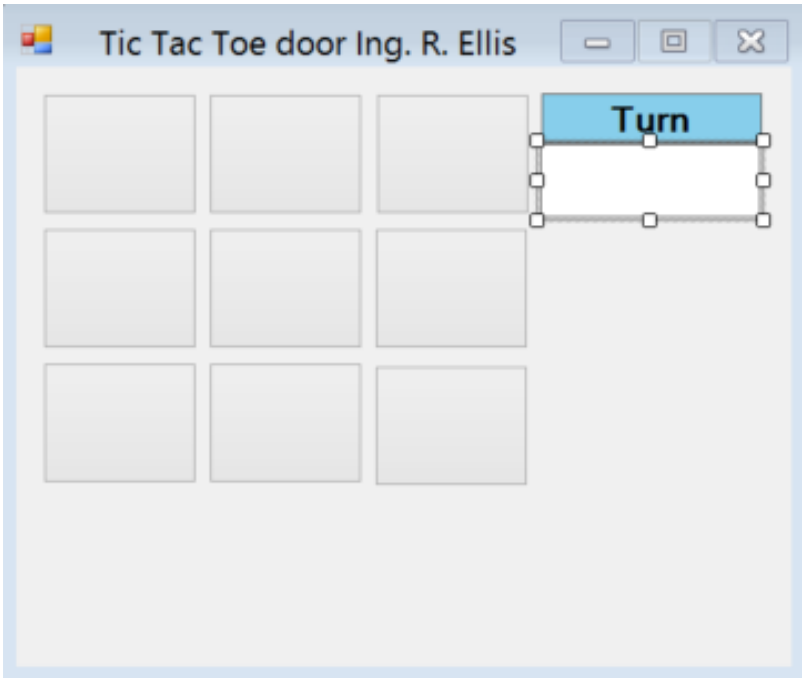
De volgende stap is om een label toe te voegen aan de form.



De eigenschappen voor de label zijn:

BackColor: {Web} – SkyBlue
BorderStyle: FixedSingle
Text: Turn
Location: 356, 18
Margin: 4, 0, 4, 0
Size: 149, 34

Het venster ziet er nu zo uit:



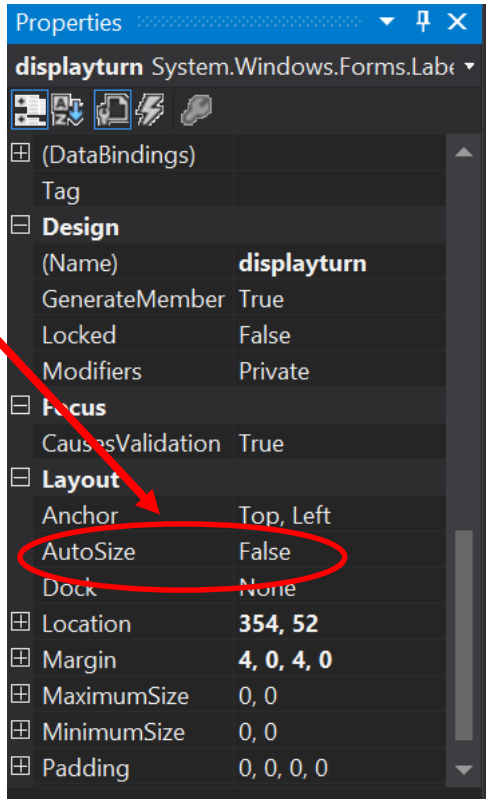
Voeg een andere label toe onder de net gemaakte label. Pas de volgende eigenschappen aan:

BackColor: {Custom} – White
ForeColor: {Custom} – Black
Text: Player 1
TabIndex: 10
(Name): displayturn
Location: 356, 52
Margin: 4, 0, 4, 0
Size: 149, 50

Doe nu hetzelfde als hierboven om de scorekaart voor speler 1 en speler 2 te maken. Geef de scorekaarten de namen player1score en player2score.

Opmerking:

Wanneer een label component gebruikt wordt zorg er voor dat de eigenschap **AutoSize** de **waarde False** heeft. Anders past Visual Studio automatisch de grootte van de label aan naar gelang wat er in de eigenschap Tekst van de label is ingevoerd.



Eigenschappen voor speler labels:

BackColor:{Web}–Green
BorderStyle: FixedSingle
ForeColor: White
Text: Player 1
Location: 356, 109
Margin: 4, 0, 4, 0
Size: 74, 34

Doe hetzelfde voor Player 2 en wijzig de Locatie naar: 429, 109

De eigenschappen voor de scorekaart is als volgt:

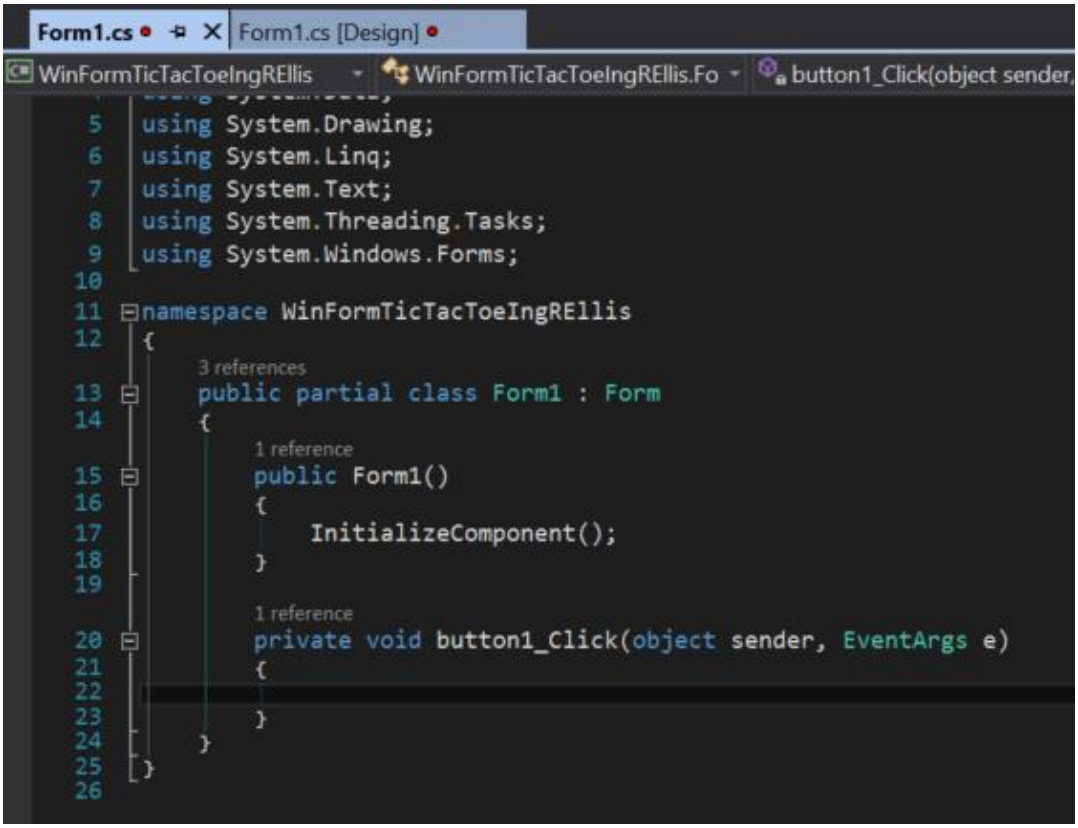
BackColor: Web}–White
BorderStyle: FixedSingle
ForeColor: White
Text: { leeg maken }
(**Name):** playe1score
Location: 356, 143
Margin: 4, 0, 4, 0
Size: 74, 57

Windows Forms

Het spel wordt gemaakt voor twee spelers, wanneer speler 1 het symbool "X" gebruikt en speler 2 het symbool "O". Moet er dus een "X" symbool op de knop komen te staan als speler 1 er op drukt en een "O" symbool wanneer dat speler 2 is.

Daar programmeren in Windows Forms Event Driven is moet het programma dus een actie uitvoeren als er op de knop gedrukt wordt. De simpelste manier om dit te doen is door te dubbel klikken op button1.

Visual studio maakt dan automatisch de Method aan voor dit event. Er moet nu nog de code in dat event getypt worden.



Herhaal deze actie voor alle 8 andere knoppen.

Voordat we de code in de events van de knoppen kunnen invoeren moeten er eerst een aantal variabelen en methods gemaakt worden.

Net onder de sluit accolade van de Method Form1 moet de volgende code worden toegevoegd:

```
public Form1()
{
    InitializeComponent();
}

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;
```

De variabele turn wordt gebruikt om te bepalen of er een “X” of een “O” geplaatst moet worden. Dit wordt gedaan door te kijken naar de volgende expressie (turn % 2 != 0) Als de expressie waar (True) is dan wordt er een “X” geplaats. Is het resultaat van de expressie niet waar (False) dan wordt her een “O” in de knop getoond.

De variabelen click1, click2, … click9 worden gebruikt om te bepalen of een knop al eerder ingedrukt. Dan mag het symbool niet veranderd worden. Je ziet dit in de code die in elke knop zijn click event wordt geschreven.

De variabelen player1 en player2 worden gebruikt om bij te houden hoeveel keer een speler het spel gewonnen heeft.

De Method display()

Deze method moet er voor zorgen dat in de label displayturn wordt aangegeven welke speler aan de beurt is.

Schrijf de volgende code meteen na de sluit accolade van de button1_Click event method.

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

De Method cleargame()

Er moet ook een method gemaakt worden om het scherm weer leeg te maken zodat er opnieuw een spel gespeeld kan worden.

De code voor deze method is als hieronder. Voeg deze code toe onder die van de method display().

```
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;
}
```

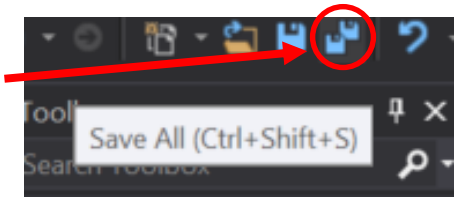
De method button1_Click()

Nu kun je de code voor de gebeurtenis invoeren, voor wanneer een speler op een knop drukt.

```
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();
}
```

Deze code is hetzelfde voor de andere 8 knoppen. Houd er wel rekening mee dat click1 en de objectnaam button1 gewijzigd dienen te worden voor de andere buttons in click2 en button2, etc.

Dubbel klik de knoppen een voor een om Visual Studie de Event Methods van de knoppen te generen. Je kunt de methods verplaatsen dat ze net onder die van de eerste knop komen te staan.
Kopieer dan de code van de eerste knop en plak het in de volgende knop en pas de namen van de variabelen click1 en button1 aan. Doe dit voor elke knop.

Denk eraan, sla je werk regelmatig 

op.

De Method checkit()

Deze methode wordt gebruikt om te controleren of een speler gewonnen heeft. Dit moet voor alle verschillende mogelijkheden gebeuren. De code is voor elke situatie hetzelfde. Alleen de condities, die de spel situatie bepalen, zijn iedere keer anders.

Let dus goed op bij het wijzigen van de objectnamen button1, button2, etc. Copy & paste en pas de code aan.

```
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();
```

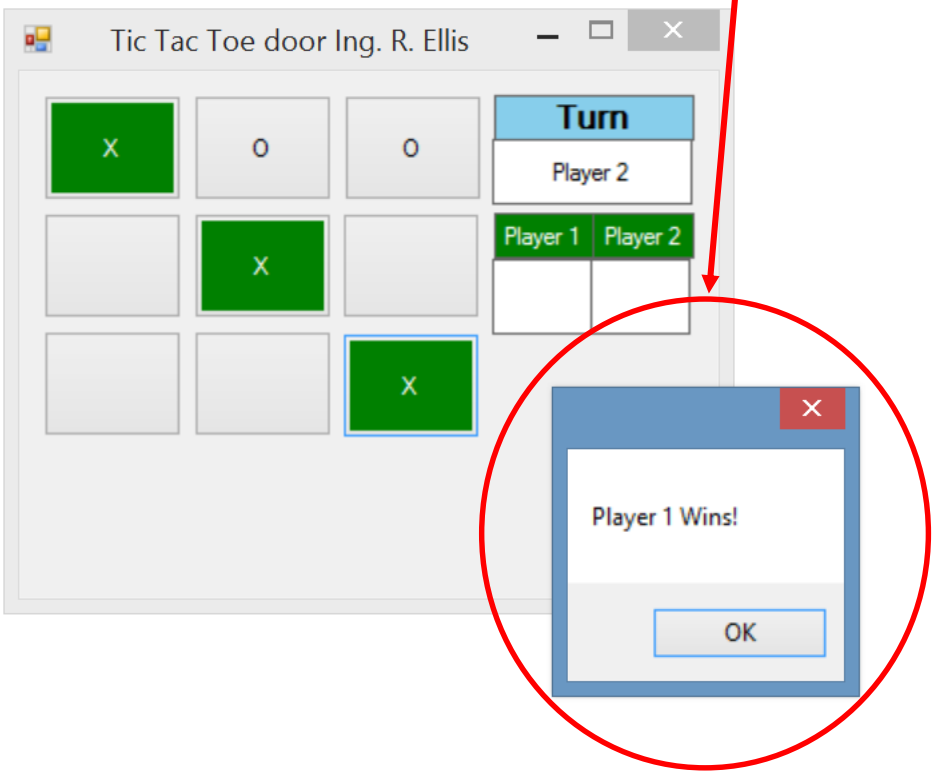
```
            }
            else
            {
                MessageBox.Show("Player 2 Wins!");
                player2++;
                player2score.Text = player2.ToString();

            }
            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();
        }
    }
}
```

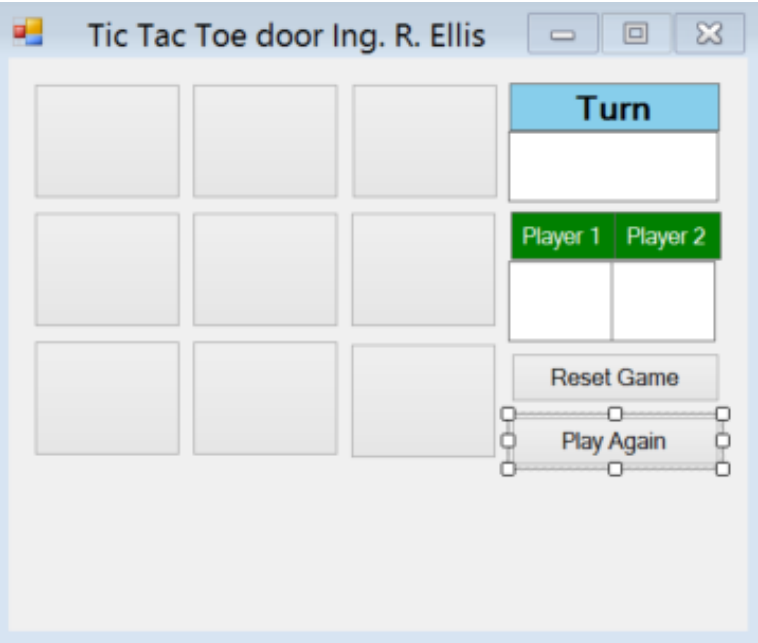
De code hieronder zorgt ervoor dat er een dialoogvenster getoond wordt dat aangeeft welke speler gewonnen heeft. Dit is een standaard component van Windows. Wanneer de gebruiker op de OK knop drukt, verdwijnt het venster en kan de gebruiker weer verder met het programma.

```
MessageBox.Show("Player 1 Wins!");
```



Windows Forms

Het programma is nu bijna klaar. Er moeten nog twee knoppen gemaakt worden. Eén voor het herstarten en een andere voor het opnieuw kunnen spelen ingeval er geen winnaar is.



Eigenschappen:
Reset Game knop:

Tekst: Reset Game
(Name): reset
Location: 356, 209
Margin: 4, 5, 4, 5
Size: 148, 35

Eigenschappen: Play Again

Tekst: Play Again
(Name): playagain
Location: 356, 254
Margin: 4, 5, 4, 5
Size: 148, 35

Nu moet nog de code geschreven worden voor om een actie te ondernemen wanneer de speller één van de knoppen aan klikt. Denk eraan dubbel klik op de knop zodat Visual Studio de methode genereert voor dit event.

De code voor de methode reset_Click() is als volgt:

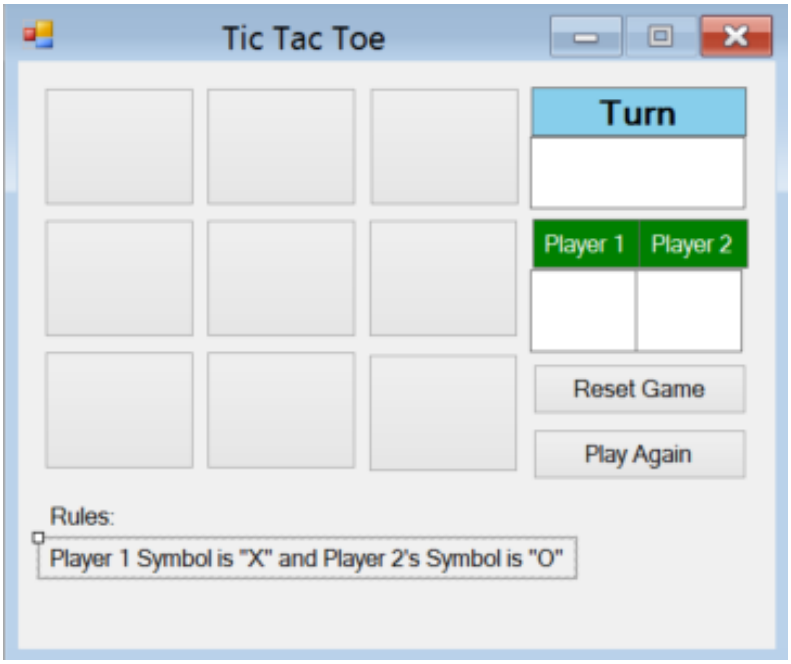
```
private void reset_Click(object sender, EventArgs e)
{
    player1score.Text = "";
    player2score.Text = "";
    player1 = 0;
    player2 = 0;
    cleargame();
}
```

De code voor de methode playagain_Click() is als volgt:

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

Om het programma helemaal gebruikersvriendelijk te maken voegen we nog twee labels onderaan toe waar in de spelregels worden uitgelegd.

De form ziet er dan als volgt uit:



Deze les heeft je de basis kennis gegeven om zelf een Windows Form programma te kunnen maken.

Veel plezier met spelen en het maken van eigen Windows Forms programma's en spelletjes.