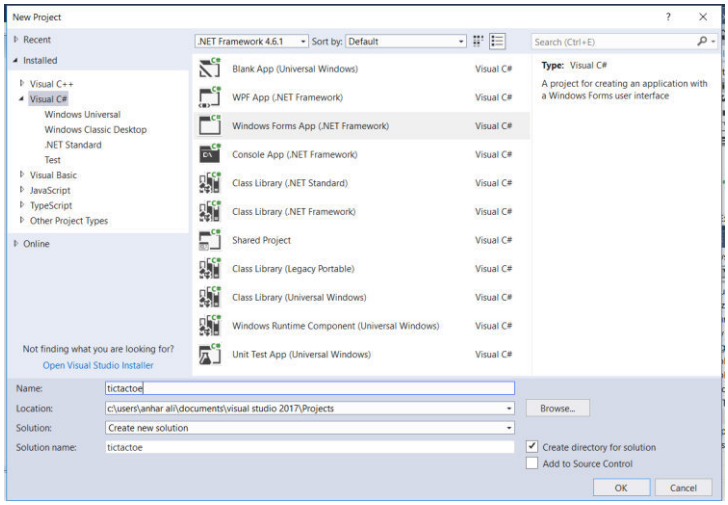
C# Tutorial – Create a Tic Tac Toe game and play against AI Opponent

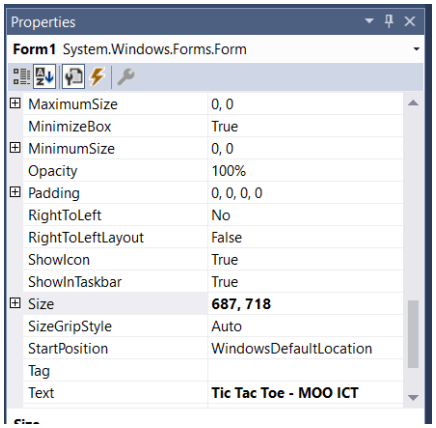
Start Visual Studio →



Create a new project in visual studio. Under the C# programming language chose Windows Form Application. Name this project tictactoe and click OK.

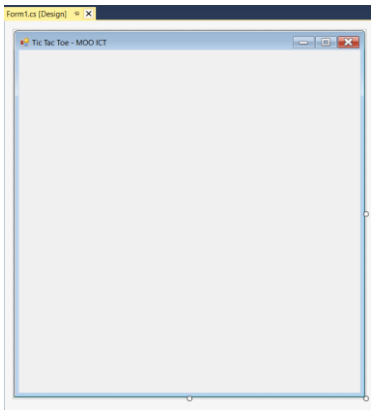
This solution will be stored inside your documents folder and then inside the Visual studio 2017 folder and then projects folder.

In the new form make the following changes to its properties



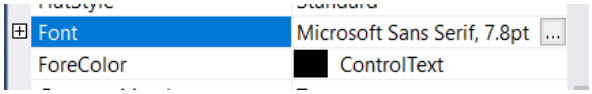
Change the size to – **687, 718**

In the text section add – **Tic Tac Toe MOO ICT**

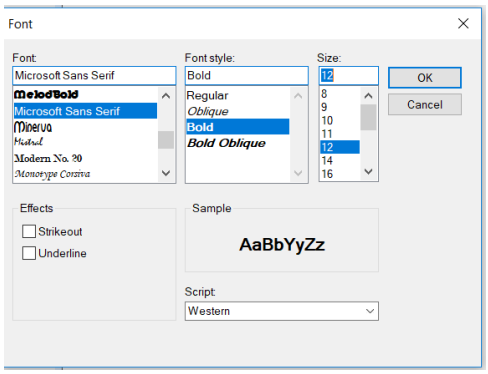


This will be final result of the form. Now we will need to add components to the form. From the tool box drag and drop two labels to the form.

Place them both on either top side of the form, these two will show the player score and computer/AI score. Let make some changes to the properties for both of these labels.



Under the font section in the properties click on the … three dotted button and it will open then font dialog box.



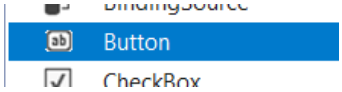
Make the font BOLD and size 12 Click OK.

Change label1 text to Player Wins – 0 and change label2 text to AI Wins – 0

Change the label1 fore colour to green and label2 fore colour to red.

Above is the final view of the labels on screen.

Now we need to add lots of buttons to the screen. Lets start with only 1. We will change its properties and then we will copy and paste it as we need it.



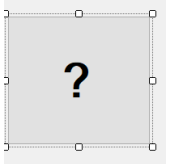
Drag and drop a button to the form.

Change the following to its properties.

Size – **177, 160**

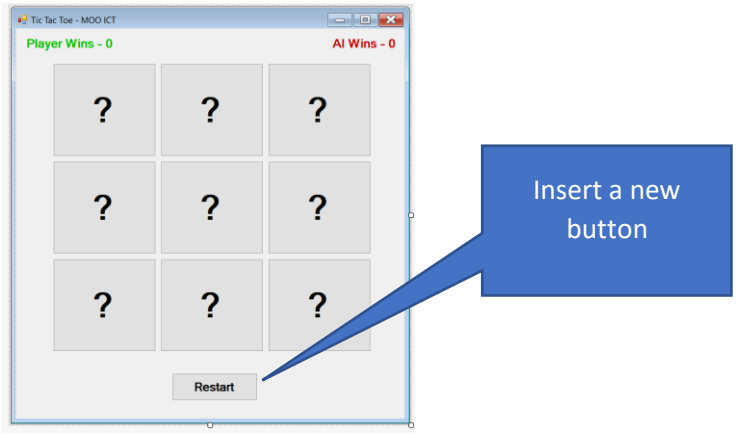
Tag – **play**

Text – **Microsoft Sans Serif, 36pt, style Bold**



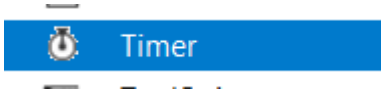
This is what the button should look like.

Now you can copy and paste the button 8 times to make the lay out look like this.

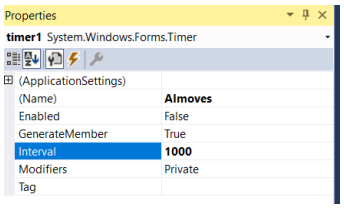


Add one more button to the scene called and change text on the button to say Restart. This button will be used to reset all of the other buttons. **Don't add any tags to the restart button.**

We need to add a Timer component to this form. Drag and drop a timer from the tool box to the form.



Make the following changes to the timer in the properties window. Change the name to **AImoves** and interval to 1000.

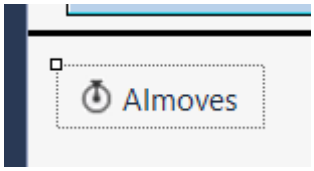


Now we need to start adding the events for this game. Select all the question mark buttons on the form. And got to then events window.

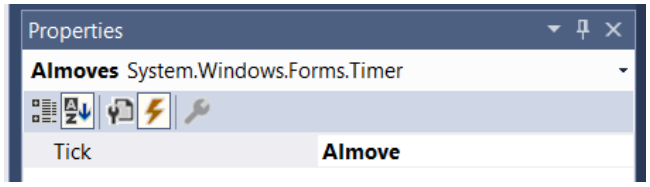

While all the buttons are highlighted, click on the lightning bolt icon in the properties window this will take you the events window.

In the events window you will find the click event, type in playerClick and press enter.

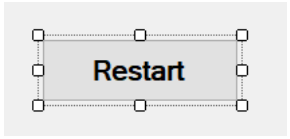
Now click on the AImoves timer object.



Same as before, In the events windows type in AImove and press enter. Notice that in the name of the timer there a S in the end but in the name of the event there is no S its just AImove.



Now click on the Restart button



In the events window on the CLICK event type restartGame and press enter.

Now we have all of our events in place time to add the code.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace tictactoe
{
    public partial class Form1 : Form
```

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

        private void playerClick(object sender, EventArgs e)
        {

        }

        private void AImove(object sender, EventArgs e)
        {

        }

        private void restartGame(object sender, EventArgs e)
        {

        }
    }
}
```

The green lines that start with the // are comments. They don't affect the code any way but it's a good way to stay organised and its usually seen as a good practise when programming. Add the highlighted code from below in to the code view. Make sure you do all the curly brackets for the functions.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace tictactoe
{
    public partial class Form1 : Form
    {

        // below is the player class which has two value
        // X and O
        // by doing this we can control the player and AI symbols
        public enum Player
        {
            X, O
        }

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

        private void playerClick(object sender, EventArgs e)
        {

        }

        private void AImove(object sender, EventArgs e)
        {

        }

        private void restartGame(object sender, EventArgs e)
        {

        }

        private void loadbuttons()
        {
```

```
        }

        private void resetGame()
        {

        }

        private void Check()
        {

        }
    }
}
```

The public enum Player is a class that we are adding this code, which contains two value X and O. This will be used to switch the players symbols in the game.

That resertGame() line added inside the Public Form1 () function will run as soon as the game start so the players can start playing.

Then you can see the 3 other function in the bottom, loadbutton, resetGame and check. They all have their own purpose in this game. Lets move further and you will see how we use them.

Variables –

```
        Player currentPlayer; // calling the player class

        List<Button> buttons; // creating a LIST or array of buttons
        Random rand = new Random(); // importing the random number generator class
        int playerWins = 0; // set the player win integer to 0
        int computerWins = 0; // set the computer win integer to 0
```

As you can see all of the variables are commented through, we will still provide some insights into this so you can better understand the process.

The first line Player currentPlayer; is calling the player class we created earlier with X and O value.

List<Button> buttons; is a list item we are creating for this game. This is a type of array where we will add all of the buttons so the AI can pick them when we play the game.

Random rand = new Random(); This will allow us to generate a random number. We can use this to pick a random button on the screen when the AI is playing the game.

Int playerWins and Int ComputerWins are both integers set to 0.

Load buttons function:

```
        private void loadbuttons()
        {
            // this the custom function which will load all the buttons from the form to the buttons list
            buttons = new List<Button> { button1, button2, button3, button4, button5, button6, button7,
button9, button8 };
        }
```

Above is the load buttons function. In this function we are adding all of the playable buttons to the list. Notice the button9 is the restart button and we are not adding it to the list because it doesn't need to be in the list.

Restart game button function:

```
        private void restartGame(object sender, EventArgs e)
        {
            // this function is linked with the reset button
            // when the reset button is clicked then
            // this function will run the reset game function
            resetGame();
        }
```

In the restart game event we are going to run the reset game function. That's it. When the restart button is clicked this reset function will fire up from within.

Reset game function:

```
        private void resetGame()
        {
            //check each of the button with a tag of play
            foreach (Control X in this.Controls)
            {
                if (X is Button && X.Tag == "play")
                {
                    ((Button)X).Enabled = true; // change them all back to enabled or clickable
                    ((Button)X).Text = "?"; // set the text to question mark
                    ((Button)X).BackColor = default(Color); // change the background colour to default
button colors
                }
            }

            loadbuttons(); // run the load buttons function so all the buttons are inserted back in the
array
        }
```

Above is the reset game function.

In this function we are running a for each LOOP which will check all the button with the TAG PLAY and reset them back to their normal state, making them all enable, setting the text to **?** mark and the background colour to default button colour.

After that we are running the load buttons function. Which will add all of the playable buttons to the list array.

Player click function

```
        private void playerClick(object sender, EventArgs e)
        {
            var button = (Button)sender; // find which button was clicked
            currentPlayer = Player.X; // set the player to X
            button.Text = currentPlayer.ToString(); // change the button text to player X
            button.Enabled = false; // disable the button
            button.BackColor = System.Drawing.Color.Cyan; // change the player colour to Cyan
            buttons.Remove(button); //remove the button from the list buttons so the AI can't click it either
            Check(); // check if the player won
            AImoves.Start(); // start the AI timer
        }
```

Above is the player click event. This event will fire each time the player clicks on the button.

First we are creating a local variable called button, this variable won't be accessible from outside the function, its secure and only serves a single purpose. Button will have the value of (Button)sender, this means the button that is clicked will follow the instructions stated in the function.

CurrentPlayer = Player.X; Player will always be X we are using the enum class we created earlier.

Button.Text = currentPlayer.ToString(); This will change the button text to X
Button.enabled = false; Once the button is clicked we will disable it.

Button.BackColor = System.Drawing.Color.Cyan; we are changing the players colour to Cyan of the clicked button.
Button.remove(button); Remember when we added all the buttons to the list, now to keep the AI from making mistakes such as clicking on disabled button or clicking on the players button we will remove the said button from the list. This way we can ensure the AI doesn't over lap with the player.

Then we are running the check function and then we are starting the AImoves timer.

Check game function:

```
        private void Check()
        {
            //in this function we will check if the player or the AI has won
            // we have two very large if statements with the winning possibilities
            if (button1.Text == "X" && button2.Text == "X" && button3.Text == "X"
               || button4.Text == "X" && button5.Text == "X" && button6.Text == "X"
               || button7.Text == "X" && button9.Text == "X" && button8.Text == "X"
               || button1.Text == "X" && button4.Text == "X" && button7.Text == "X"
```

```
                || button2.Text == "X" && button5.Text == "X" && button8.Text == "X"
                || button3.Text == "X" && button6.Text == "X" && button9.Text == "X"
                || button1.Text == "X" && button5.Text == "X" && button9.Text == "X"
                || button3.Text == "X" && button5.Text == "X" && button7.Text == "X")
            {
                // if any of the above conditions are met
                AImoves.Stop(); //stop the timer
                MessageBox.Show("Player Wins"); // show a message to the player
                playerWins++; // increase the player wins
                label1.Text = "Player Wins- " + playerWins; // update player label
                resetGame(); // run the reset game function
            }
            // below if statement is for when the AI wins the game
            else if (button1.Text == "O" && button2.Text == "O" && button3.Text == "O"
            || button4.Text == "O" && button5.Text == "O" && button6.Text == "O"
            || button7.Text == "O" && button9.Text == "O" && button8.Text == "O"
            || button1.Text == "O" && button4.Text == "O" && button7.Text == "O"
            || button2.Text == "O" && button5.Text == "O" && button8.Text == "O"
            || button3.Text == "O" && button6.Text == "O" && button9.Text == "O"
            || button1.Text == "O" && button5.Text == "O" && button9.Text == "O"
            || button3.Text == "O" && button5.Text == "O" && button7.Text == "O")
            {

                // if any of the conditions are met above then we will do the following
                // this code will run when the AI wins the game
                AImoves.Stop(); // stop the timer
                MessageBox.Show("Computer Wins"); // show a message box to say computer won
                computerWins++; // increase the computer wins score number
                label2.Text = "AI Wins- " + computerWins; // update the label 2 for computer wins
                resetGame(); // run the reset game
            }
        }
```

Above is the check function. After each move is made this function will run to check if the computer or the player won.

There are two BIG if statements in this function. The two || symbols mean OR in if statement for example

```
If (button1.text == "X" && button2.text == "X" || button1.text == "O" && button2.text == "O")

{

Do something here

}
```

If button 1 and 2 text are X OR button 1 and button 2 text are O then this if statement will do something.

When once of statements are true then we will stop the timer show a message that either the computer or the player won the game and then we will add 1 to the player or computers score and finally reset the game so they can play again.

All the winning combinations for player X and O are made in this if statement.

Look through this function to make sure you understand it fully.

AImove function:

```
        private void AImove(object sender, EventArgs e)
        {
            // THE CPU will randomly choose a button from the list to click.
            // While the array is greater than 0 the cpu will operate in the game
            // if the array is less than 0 it will stop playing
            if (buttons.Count > 0)
            {
                int index = rand.Next(buttons.Count); // generate a random number within the number of
buttons available
                buttons[index].Enabled = false; // assign the number to the button
                // when the random number is generated then we will look into the list
                // for what that number is we select that button
                // for example if the number is 8
                // then we select the 8th button in the list

                currentPlayer = Player.O; // set the AI with O
                buttons[index].Text = currentPlayer.ToString(); // show O on the button
                buttons[index].BackColor = System.Drawing.Color.DarkSalmon; // change the background of
the button dark salmon colour
                buttons.RemoveAt(index); // remove that button from the list
```

```
                Check(); // check if the AI won anything
                AImoves.Stop(); // stop the AI timer
            }
        }
```

In the function we start the if statement, we cannot run the AI if there is no button to play on. So we are checking if the button list has at least one button available then we follow the next instruction.

We are creating a local integer called index this we generate a random number depending on the buttons in the button list. We can count all of the buttons in the list by calling the Count property in the buttons list.

Then we are setting the AI to player.O

We will change the button text to the O

Change the back colour of the button played by the AI to Dark Salmon

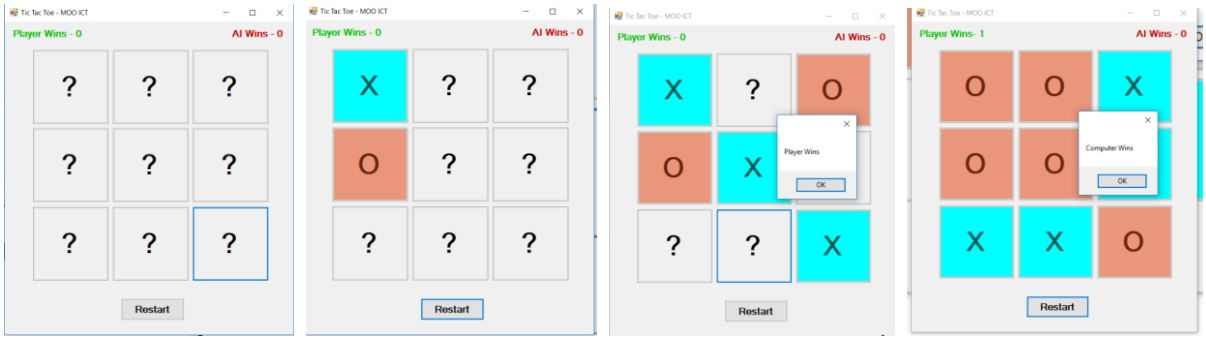
We will remove the button played by the AI from the buttons list

Check the game so see if there is any condition that was met by this move.

Stop the timer once the moves are made.

Now lets run the game to see where we are with this all this

Click the debug button or press the F5 button to runt the project in visual studio.



Full Code for the game –

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace tictactoe
{
    public partial class Form1 : Form
    {

        // below is the player class which has two value
        // X and O
        // by doing this we can control the player and AI symbols
        public enum Player
        {
            X, O
        }

        Player currentPlayer; // calling the player class
```

```
        List<Button> buttons; // creating a LIST or array of buttons
        Random rand = new Random(); // importing the random number generator class
        int playerWins = 0; // set the player win integer to 0
        int computerWins = 0; // set the computer win integer to 0

        public Form1()
        {
            InitializeComponent();
            resetGame(); // call the set game function
        }

        private void playerClick(object sender, EventArgs e)
        {
            var button = (Button)sender; // find which button was clicked
            currentPlayer = Player.X; // set the player to X
            button.Text = currentPlayer.ToString(); // change the button text to player X
            button.Enabled = false; // disable the button
            button.BackColor = System.Drawing.Color.Cyan; // change the player colour to Cyan
            buttons.Remove(button); //remove the button from the list buttons so the AI can't
click it either
            Check(); // check if the player won
            AImoves.Start(); // start the AI timer
        }

        private void AImove(object sender, EventArgs e)
        {
            // THE CPU will randomly choose a button from the list to click.
            // While the array is greater than 0 the cpu will operate in the game
            // if the array is less than 0 it will stop playing
            if (buttons.Count > 0)
            {
                int index = rand.Next(buttons.Count); // generate a random number within the
number of buttons available
                buttons[index].Enabled = false; // assign the number to the button
                // when the random number is generated then we will look into the list
                // for what that number is we select that button
                // for example if the number is 8
                // then we select the 8th button in the list

                currentPlayer = Player.O; // set the AI with O
                buttons[index].Text = currentPlayer.ToString(); // show O on the button
                buttons[index].BackColor = System.Drawing.Color.DarkSalmon; // change the
background of the button dark salmon colour
                buttons.RemoveAt(index); // remove that button from the list
                Check(); // check if the AI won anything
                AImoves.Stop(); // stop the AI timer
            }
        }

        private void restartGame(object sender, EventArgs e)
        {
            // this function is linked with the reset button
            // when the reset button is clicked then
            // this function will run the reset game function
            resetGame();
        }

        private void loadbuttons()
        {
            // this the custom function which will load all the buttons from the form to the
buttons list
            buttons = new List<Button> { button1, button2, button3, button4, button5, button6,
button7, button9, button8 };
        }

        private void resetGame()
        {
            //check each of the button with a tag of play
            foreach (Control X in this.Controls)
            {
                if (X is Button && X.Tag == "play")
                {
```

```
                    ((Button)X).Enabled = true; // change them all back to enabled or clickable
                    ((Button)X).Text = "?"; // set the text to question mark
                    ((Button)X).BackColor = default(Color); // change the background colour to
default button colors
                }
            }

            loadbuttons(); // run the load buttons function so all the buttons are inserted
back in the array
        }

        private void Check()
        {
            //in this function we will check if the player or the AI has won
            // we have two very large if statements with the winning possibilities
            if (button1.Text == "X" && button2.Text == "X" && button3.Text == "X"
               || button4.Text == "X" && button5.Text == "X" && button6.Text == "X"
               || button7.Text == "X" && button9.Text == "X" && button8.Text == "X"
               || button1.Text == "X" && button4.Text == "X" && button7.Text == "X"
               || button2.Text == "X" && button5.Text == "X" && button8.Text == "X"
               || button3.Text == "X" && button6.Text == "X" && button9.Text == "X"
               || button1.Text == "X" && button5.Text == "X" && button9.Text == "X"
               || button3.Text == "X" && button5.Text == "X" && button7.Text == "X")
            {
                // if any of the above conditions are met
                AImoves.Stop(); //stop the timer
                MessageBox.Show("Player Wins"); // show a message to the player
                playerWins++; // increase the player wins
                label1.Text = "Player Wins- " + playerWins; // update player label
                resetGame(); // run the reset game function
            }
            // below if statement is for when the AI wins the game
            else if (button1.Text == "O" && button2.Text == "O" && button3.Text == "O"
            || button4.Text == "O" && button5.Text == "O" && button6.Text == "O"
            || button7.Text == "O" && button9.Text == "O" && button8.Text == "O"
            || button1.Text == "O" && button4.Text == "O" && button7.Text == "O"
            || button2.Text == "O" && button5.Text == "O" && button8.Text == "O"
            || button3.Text == "O" && button6.Text == "O" && button9.Text == "O"
            || button1.Text == "O" && button5.Text == "O" && button9.Text == "O"
            || button3.Text == "O" && button5.Text == "O" && button7.Text == "O")
            {

                // if any of the conditions are met above then we will do the following
                // this code will run when the AI wins the game
                AImoves.Stop(); // stop the timer
                MessageBox.Show("Computer Wins"); // show a message box to say computer won
                computerWins++; // increase the computer wins score number
                label2.Text = "AI Wins- " + computerWins; // update the label 2 for computer
wins
                resetGame(); // run the reset game
            }
        }
    }
}
```