C# Tutorial Create a Rock Paper and Scissors Game

Welcome to this exciting new tutorial. Today we will create a simple rock paper and scissor game in Visual Studio and C#. It's going to be a very straight forward tutorial for you to follow. We will be creating a simple AI – Artificial intelligence for this game which will play alongside the user and determine whether you won or lost a round.

Lesson objectives –

1. Create simple GUI interface for rock, paper and scissors game
2. Use picture boxes to load various images for the game
3. Use button events
4. Use timer event / start and stop timer event according to the requirement
5. Calculating each player rounds
6. Determine who won the game
7. Input player Name
8. Use that name to display on the form
9. Use visual basic framework inside of C# - make the different programming language communicate with each other
10. Show help screen form for user accessibility

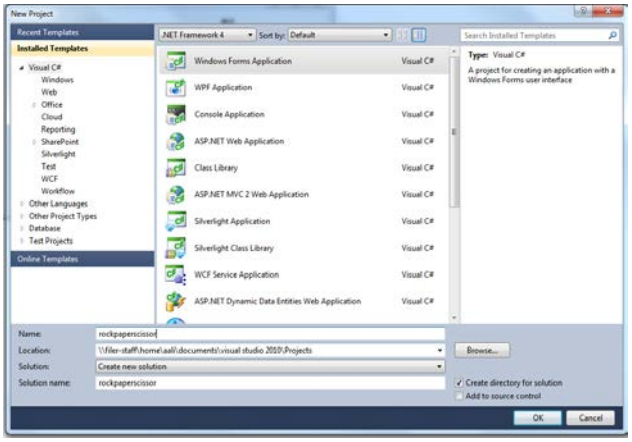
All the images used in this game are available on **MOOICT.COM** and can be downloaded from there. If you want to use your own images then feel free to use them. Make sure to follow the tutorial step by step to prevent any errors along the way.

First let's create a new project in visual studio

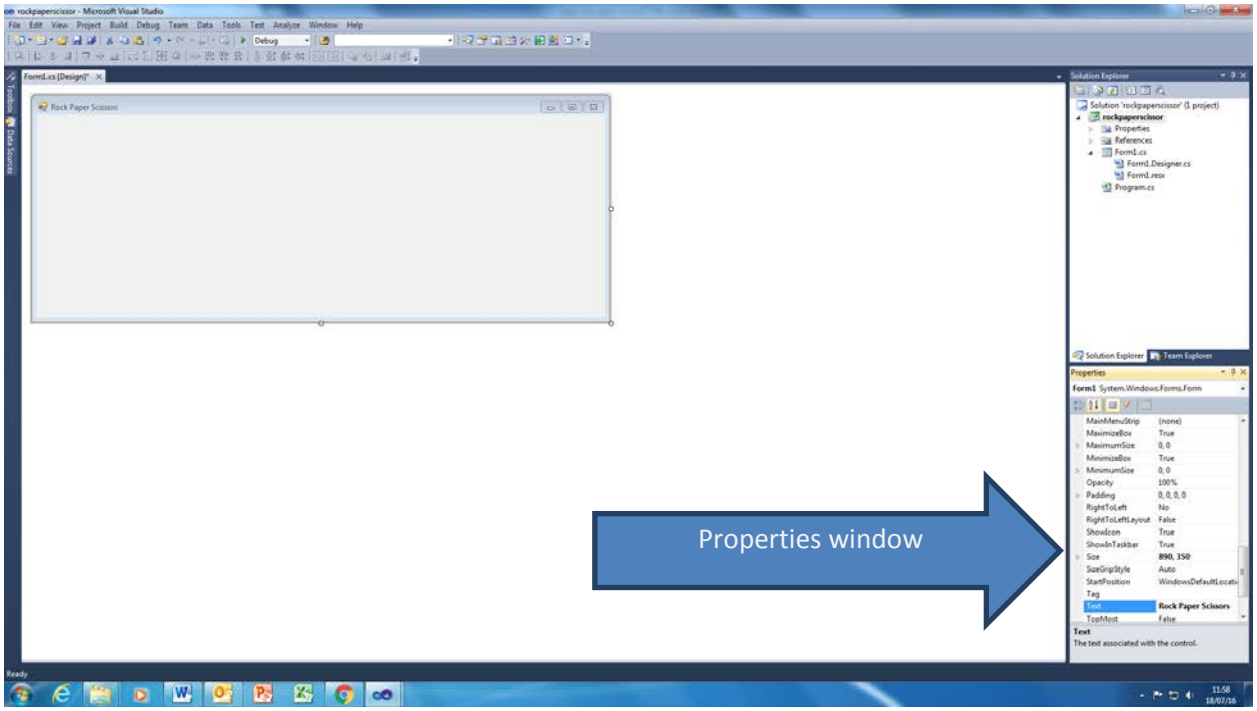
C# -> Windows Form Application

Name it rockpaperscissors

Click OK



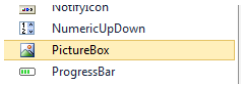
First thing to do will be we need to change the title of the form from Form1 to Rock Paper Scissors and change the size of the form from 300,300 to **890**, **350**. If you can't see the properties window right click on the form and click on properties.



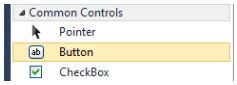
We will need that space to make sure our user can see the AI and the labels we will place on the form.

Thing we will need for this project

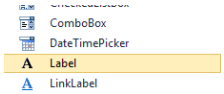
2 picture boxes



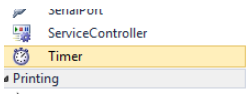
4 buttons



6 labels



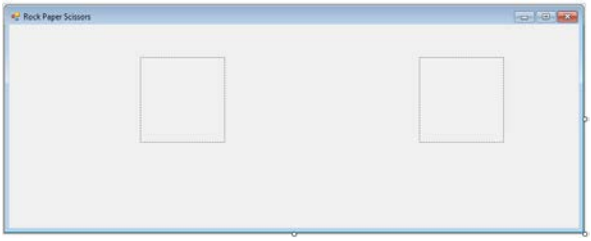
1 timer object



All of these components can be found in the tool box. If you can't see the tool box then goto view - > tool box or press CTRL+W+X this will bring up the tool box.

Let's place the items on screen then change some properties.

Firstly place two of the picture boxes on screen as below



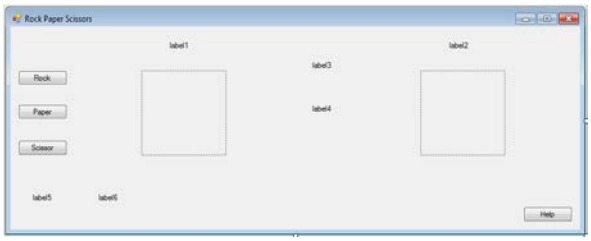
Change the size of the picture boxes to 130, 130 meaning 130 height x 130 width

Now add the 4 buttons to the screen

Change the text accordingly

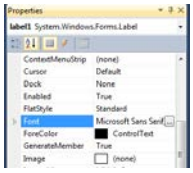
- Button1 = Rock
- Button2 = Paper
- Button3 = Scissor
- Button4 = Help

Now we will add the labels on the screen. In visual studio it has a built in setting where we can change the font, colour and the size to make it look like the way we want to. Let's add all the labels on the screen then we will change the settings accordingly.



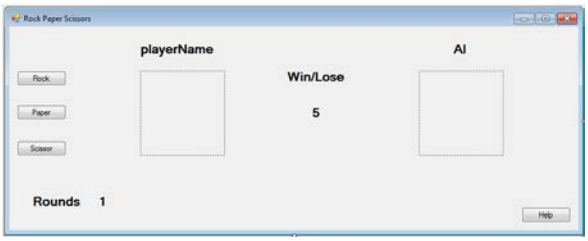
Here are the 6 labels on the screen.

Now click on label1 and check the properties for the option called – Font – Click on the 3 dots …



Change to the following setting – Font Microsoft Sans Serif – Style Bold – Size 14

Now do it for all of the labels on the screen



Here I have changed the text of each label. Let's discuss the purpose of them all now.

Player name label will contain the name of the player once the program asks the player to enter their names in the box that we will sort out further down.

AI will only say AI because it doesn't need a specific name.

Win/Lose is the label which will show who won the game either the player or the AI

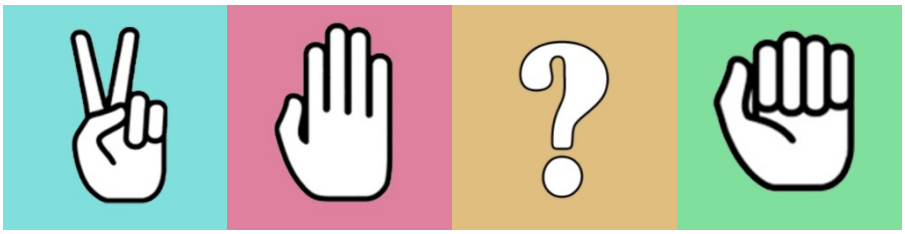
Label that's showing 5 on the text is going to be our count down timer. We will give the player 5 seconds to choose rock paper or scissor. Once the countdown goes to 0 we will reveal the answer.

Rounds will only show Rounds it don't need to change at any time in the game

The label showing 1 will show how many rounds we have played so far. This game will have 3 rounds to player and then it will determine the final winner of the game.

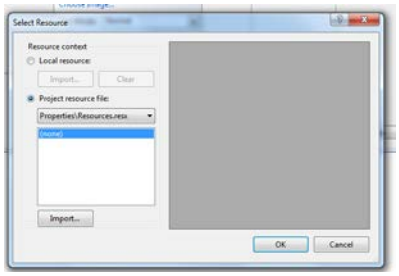
Now lets load the images to the project.

Here are the images I prepared for this project. You can use your own if you want.



Now click on the picturebox1 and click on the little triangle on the top right corner of it.

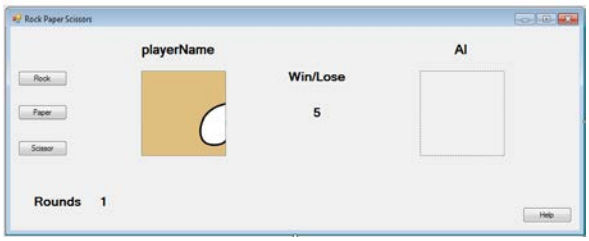
Click on choose image



Click on Import to move the pictures inside the project

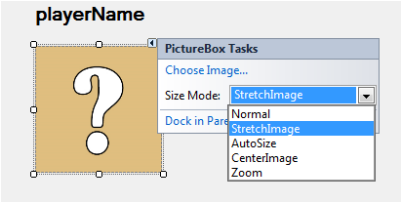
Select all and then click on OPEN

Now click on OK



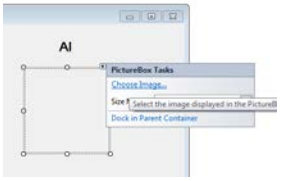
As you can see the image is not set up properly because we can't see the question mark. We need to change the setting on the image so we can see the whole image instead of this.

Click on picturebox1 and click on the little triangle again.

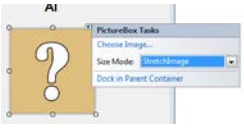


Change the size mode to Stretch Image, this will show the image according to the size of the picture.

For picture box 2 you won't need to import them again you simply pick the question mark from the resources window.

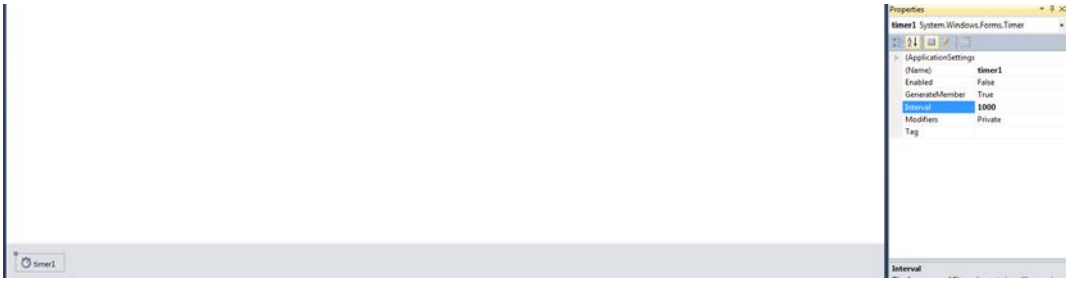


Click OK



Change the size to stretch image again and you are all done.

Now we will need to add the timer to this project.



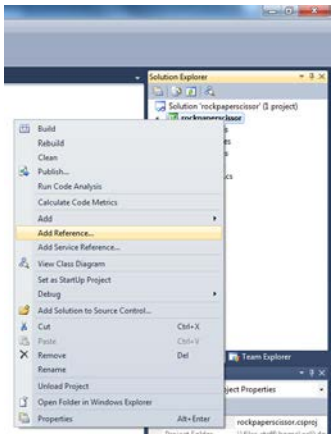
Change the interval in the properties window to 1000. The timer event calculates time in the milliseconds so in order to get the right behaviours from the program we will need the change it to 1000 so it calculates in seconds not milliseconds.

Now we can get started on the programming on this project.

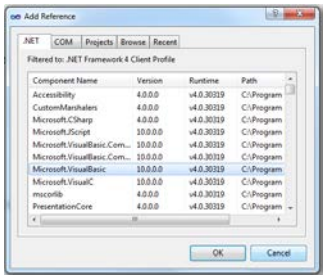
Before we get started on the programming, I briefly mentioned in the lesson objectives earlier that we will be communicating with the Visual Basic programming directly from C#. Now although Visual Basic is an older language and only used in the industry focused purpose we still can use for our own needs when we want to. It's a nice little touch from Microsoft. Thank you.

So for this project we will be adding a reference from Visual Basic programming in to C#. Here is how we do it.

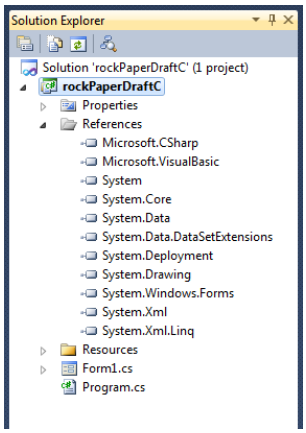
Right click on the project solution folder directly on the rockpaperscissors



Click on Add Reference



Find the Microsoft.VisualBasic and click on OK.

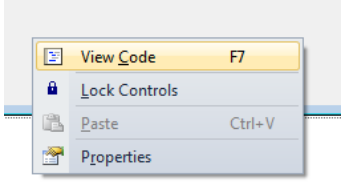


Now there is a reference to visual basic has been added inside this project.

Why are we doing this?

Well we need to capture the user name for this project, In C# there is no easier way to do this except creating several different forms and then getting them to communicate with other. In overall terms that's a very good way to do it, but we are looking for an easier way and a way that we can learn something new in this concept.

With that being done now we can continue with programming. Right click on the Form and click on View code or Press F7.




```
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
```

```
usingSystem.Data;
usingSystem.Drawing;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Windows.Forms;

namespacerockpaperscissor
{
publicpartialclassForm1 : Form
    {
public Form1()
        {
InitializeComponent();
        }
    }
}
```

This is the code you will see inside the code view.

Everything we are going to input will go under the line **public partial class Form1: Form** makes sure you don't delete any of the curly brackets because those are important.

Now the first line we will input in this program will be under the line InitializeComponent();

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

namespacerockpaperscissor
{
publicpartialclassForm1 : Form
    {
public Form1()
        {
InitializeComponent();
String input = Microsoft.VisualBasic.Interaction.InputBox("Enter Your Name", "Rock Paper and Scissor Game", "..", 0, 0);
        }
    }
}
```

We are declaring a String called input. Which will contain the reference from Visual basic and this will show a input box asking the player to enter their name. Here is the break down of the code

`Microsoft.VisualBasic.Interaction.InputBox()` – this part is calling the visual basic input box to appear on screen.

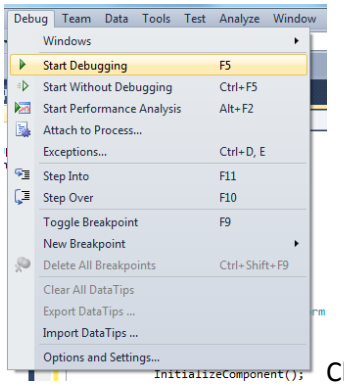
Inside the brackets we enter the question first which is "Enter your name"

After the comma we enter the title of the dialog box which is "Rock Paper And Scissor Game"

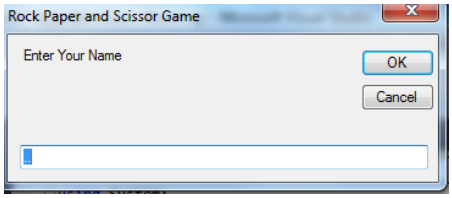
After the second comma we enter the default place holder inside the box which is 2 dots ".."

After the third comma we put the X and Y position of the box in this case its 0 for both.

Now lets run the program to see if it works.



Click on start debugging or Press F5



Nice its working

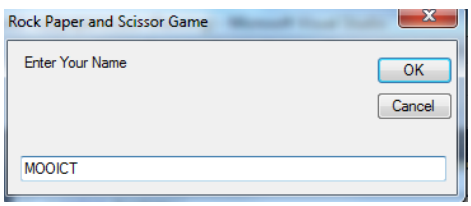
Once you click OK the full form shows up on the screen. Excellent.

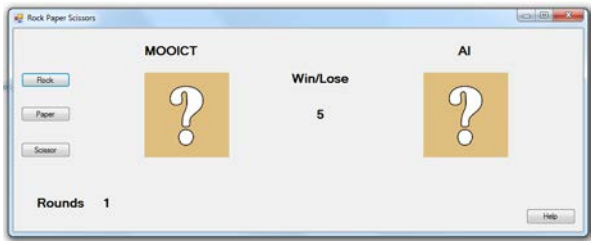
Now we need to show the player name somewhere on the screen. Lets do that.

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

namespacerockpaperscissor
{
publicpartialclassForm1 : Form
    {
public Form1()
        {
InitializeComponent();
String input = Microsoft.VisualBasic.Interaction.InputBox("Enter Your Name", "Rock Paper and Scissor
Game", "..", 0, 0);
label1.Text = input;
        }
    }
}
```

The new line is label1.text here we will view the value from the input string we captured before. Let's try it out.

Run the program, enter the name MOOICT and click OK





Hooray it's working. Feeling good aye, well you should.

So to sum it up for now we used the Visual Basic Reference to allow us to use that input form where in C# we would have to create it ourselves. Do anything that makes your life easier, even if that mean we work hard now to enjoy later.

Lets get going with rest of the programming now.

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

namespacerockpaperscissor
{
publicpartialclassForm1 : Form
    {
//declaring the variables for this game

publicint rounds = 3; // 3 rounds per game
publicinttimePerRound = 6; // 5 seconds per rounds
string[] AIchoice = { "rock", "paper", "scissor", "rock", "scissor", "paper" }; //enemy choice
options stored inside an array for easy access
publicintrandomNumber;
string command;
Randomrnd = newRandom();
stringplayerChoice;
intplayerWins = 0;
intAIwins = 0;

// end of declaring variables


public Form1()
        {
InitializeComponent();
String input = Microsoft.VisualBasic.Interaction.InputBox("Enter Your Name", "Rock Paper and Scissor
Game", "..", 0, 0);
            label1.Text = input;
        }
    }
}
```

Notice the highlighted code above.

First we are declaring a public integer called rounds which holds the value 3 inside it. Basically this will determine which round are we playing in this game.

Second integer we will be declaring is the timePerRound which will hold the number 6 inside it. Basically when the time starts it will reduce it by one straight away so we gave it a value of 6 to make start from 5.

3rd variable is a string ARRAY which holds multiple values inside it. I'm sure you are wondering why am I inputting the values multiple times. This is a small hack to make sure that the AI makes different choices. Since we will choose a Random number from 1 and 3 it will most likely make the same choice over and over so to overcome that system we need to add more values to the array and it will chose different one each time.

In this case we are stating string[] AIchoice = {"rock", "paper", "scissor ", "rock", "scissor", "paper"};

0	1	2	3	4	5
Rock	Paper	Scissor	Rock	Scissor	Paper

When we randomize this index it will show different ones instead of the same one multiple times.

Public intrandomNumber will hold the random value inside it.

String command will be used as the AI final choice in the game

Random rnd = new Random() will generate the random number for us.

String playerChoice is where we make the choice between rock, paper and scissor.

Integer player wins is where we add how many rounds the player won.

Integer AIwins is where we declare how many rounds the AI won.

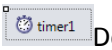
```
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingSystem.Data;
usingSystem.Drawing;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Windows.Forms;

namespacerockpaperscissor
{
publicpartialclassForm1 : Form
    {
//declaring the variables for this game

publicint rounds = 3; // 3 rounds per game
publicinttimePerRound = 6; // 5 seconds per rounds
string[] AIchoice = { "rock", "paper", "scissor", "rock", "scissor", "paper" }; //enemy choice
options stored inside an array for easy access
publicintrandomNumber;
string command;
Randomrnd = newRandom();
stringplayerChoice;
intplayerWins = 0;
intAIwins = 0;

// end of declaring variables


public Form1()
        {
InitializeComponent();
String input = Microsoft.VisualBasic.Interaction.InputBox("Enter Your Name", "Rock Paper and Scissor
Game", "..", 0, 0);
            label1.Text = input;
timer1.Enabled = true;
playerChoice = "none";
        }
```

```
    }
}
```

Add the line that says timer1.enabled = true. The timer added before we want to start it when we have entered our name into the game.

We are also setting up the playerChoice as none is beginning of the game.

Go back to the design view of the game and double click on the timer1 icon



Double click on this

```
privatevoid timer1_Tick(object sender, EventArgs e)
        {

        }
```

Visual studio will automatically create this event for the timer. Everything we want the time to do will be included inside this event.

Here is the final timer 1 tick function

```
privatevoid timer1_Tick(object sender, EventArgs e)
        {
timePerRound--; // reduce the time by 1
            label4.Text = Convert.ToString(timePerRound); // show the time on the screen
if (timePerRound< 1) // if the time is less then one second
            {
                timer1.Enabled = false; // we disable the timer if less then one second left
timePerRound = 6; // set the timer back to 6 seconds

randomNumber = rnd.Next(0, 5); // randomize the number again
                command = AIchoice[randomNumber]; // we set up the AI choice according to the random
number
// the switch statement below will show the AI choice and change the picture box images
switch (command)
                {
case"rock":
                        pictureBox2.Image = Properties.Resources.rock;
break;
case"paper":
                        pictureBox2.Image = Properties.Resources.paper;
break;
case"scissor":
                        pictureBox2.Image = Properties.Resources.scissors;
break;
default:
break;
                }
// if we have more rounds to the play then we run the check game function
if (rounds > 1)
                {
checkGame();
                }
// if we dont have any more rounds to play then we go to the final decision engine
else
                {
decisionEngine();
                }
            }
        }
```

The comments are self-explanatory.

We haven't created the **checkGame** or the **decisionEngine** function yet. Make sure you enter the code above exactly as they are.

Lets create the **checkGame()** function now

```
privatevoidcheckGame()
        {
if (playerChoice == "rock"&& command == "paper")
            {
MessageBox.Show("AI Wins");
AIwins++;
                rounds--;
nextRound();
            }
elseif (playerChoice == "paper"&& command == "rock")
            {
MessageBox.Show("player Wins");
playerWins++;
                rounds--;
nextRound();
            }
elseif (playerChoice == "paper"&& command == "scissor")
            {
MessageBox.Show("AI Wins");
AIwins++;
                rounds--;
nextRound();
            }
elseif (playerChoice == "scissor"&& command == "paper")
            {
MessageBox.Show("player Wins");
playerWins++;
                rounds--;
nextRound();
            }
elseif (playerChoice == "scissor"&& command == "rock")
            {
MessageBox.Show("AI Wins");
AIwins++;
                rounds--;
nextRound();
            }
elseif (playerChoice == "rock"&& command == "scissor")
            {
MessageBox.Show("player Wins");
playerWins++;
                rounds--;
nextRound();
            }
elseif (playerChoice == "none")
            {
MessageBox.Show(label1.Text + " Make a choice");
nextRound();
            }
else
            {
MessageBox.Show("Draw");
nextRound();
            }
        }
```

As you can see this function deals with the player and AI choice. Since the rule of the game is very simple we can use if and else if statements to ensure who won the game.

We are using multiple conditions in the if statements. In a normal is statement you have only 1 condition for example

If (doorOpen = true)

{

//go through the door

}

Else

{

//open the door etc

}

In this case we are using multiple values because this game doesn't just depend on the player choice it also depends on the AI choice so we need to determine how the player or the AI will win the game.

```
if (playerChoice == "rock"&& command == "paper")
            {
MessageBox.Show("AI Wins");
AIwins++;
rounds--;
nextRound();
            }
```

If the player chose rock and (&&) AI chose paper

Message box will show AI Wins

Add one to the AIwins variable

Deduct one from the rounds variable

Run the nextRound()function.

When the AI wins the game we are invoking the command **AIwins++** this means that each time the AI wins a round over the player we add 1 to its winning list. So if it won 1 round and then another we will add 1 to the existing 1 which will be 2.

Also we will deduct 1 from the rounds which means they can only play for 3 rounds each time.

We also needed to have a default operation because if they player doesn't make a choice in this game then we need to feedback to the player to make a choice.

```
elseif (playerChoice == "none")
            {
MessageBox.Show(label1.Text + " Make a choice");
nextRound();
            }
```

If the player didn't make a choice then the default choice will be none. Once the program sees none it will just call the players name and ask them to make a choice to play the game.

Right under the checkGame function make the decision engine function

```
privatevoiddecisionEngine()
        {
if (playerWins>AIwins)
```

```
            {
label3.Text = label1.text + " Wins the game";
            }
else
            {
label3.Text = "AI Wins the game";
            }
        }
```

This function only runs when we have played the all 3 rounds in the game. So if the player won more rounds then AI we will show the player has won. If that isn't true then the only logical choice is the AI won.

Simple and straight to the point.

Now let's create the next round function

```
privatevoidnextRound()
        {
playerChoice = "none";
            pictureBox1.Image = Properties.Resources.qq;
            timer1.Enabled = true;
            pictureBox2.Image = Properties.Resources.qq;

        }
```

This function will reset the player choice to none, reset the player/AI image to question mark and start the timer again.

Note – double the code so far in order to move to the next part.

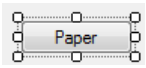
Now lets add events to the buttons Rock paper and scissor

Double click on the rock button


```
privatevoid button1_Click(object sender, EventArgs e)
        {
playerChoice = "rock";
            pictureBox1.Image = Properties.Resources.rock;
        }
```

Add the highlighted code inside the function. We are giving the player choice to rock and changing the image to the picture of rock.

Now double click on the paper button.




```
privatevoid button2_Click(object sender, EventArgs e)
        {
playerChoice = "paper";
            pictureBox1.Image = Properties.Resources.paper;
        }
```

Add the highlighted code inside the function. We are changing the player choice to paper and also changing the image to paper from the resources.

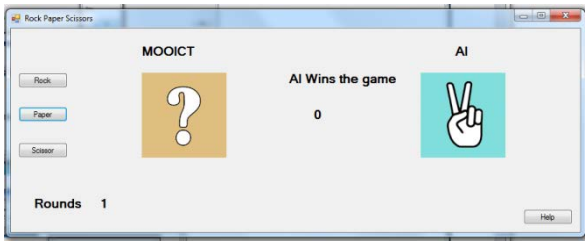
Now double click on the button scissor


```
privatevoid button3_Click(object sender, EventArgs e)
        {
playerChoice = "scissor";
            pictureBox1.Image = Properties.Resources.scissors;
        }
```

Add the highlighted code inside the function. It's the same as before only with the new scissors

Now then we can test the program to see if its running

Click on Debug -> start debugging or press F5.



The game seems to be working so far. However we need to link the rounds labels to the rounds integer and we need to add something for the help screen.

Rounds text

Inside the timer1 tick function add the following

```
privatevoid timer1_Tick(object sender, EventArgs e)
```

```
        {
label6.Text = Convert.ToString(rounds);
timePerRound--; // reduce the time by 1
            label4.Text = Convert.ToString(timePerRound); // show the time on the screen
if (timePerRound< 1) // if the time is less then one second
            {
                timer1.Enabled = false; // we disable the timer if less then one second left
timePerRound = 6; // set the timer back to 6 seconds

randomNumber = rnd.Next(0, 5); // randomize the number again
                command = AIchoice[randomNumber]; // we set up the AI choice according to the random
number
// the switch statement below will show the AI choice and change the picture box images

switch (command)
                {
case"rock":
                        pictureBox2.Image = Properties.Resources.rock;
break;
case"paper":
                        pictureBox2.Image = Properties.Resources.paper;
break;
case"scissor":
                        pictureBox2.Image = Properties.Resources.scissors;
break;
default:
break;
                }
// if we have more rounds to the play then we run the check game function
if (rounds > 1)
                {
checkGame();
                }
// if we dont have any more rounds to play then we go to the final decision engine
else
                {
decisionEngine();
                }
            }
        }
```

As you can see our rounds label is called label6 now this labels will show how many rounds we have left once the timer starts. So each time the timer starts it will show the number of rounds left to play. Try it out.

Sorted now for the Help button.

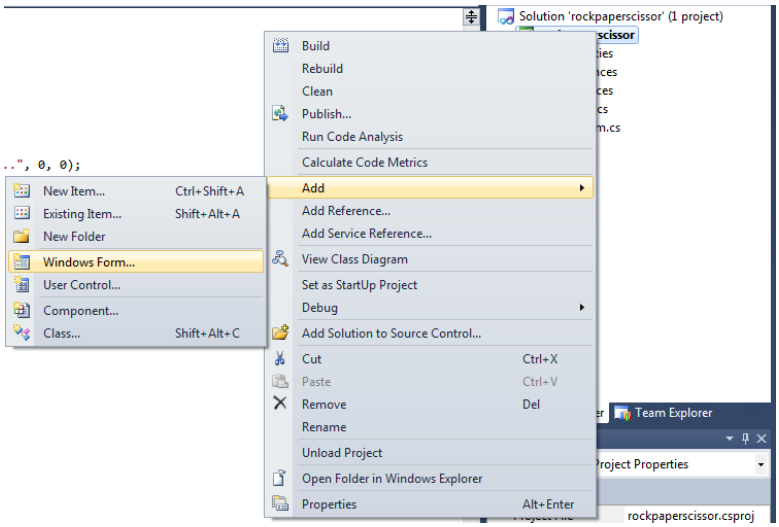
Now its important as software developers we make sure we know how to support our end user now im sure you have seen this image on the internet

	Source http://imgur.com/gallery/jlUAnFb

Although it's a joke we should always think how we can make the end user understand the software and use it efficiently without any errors.

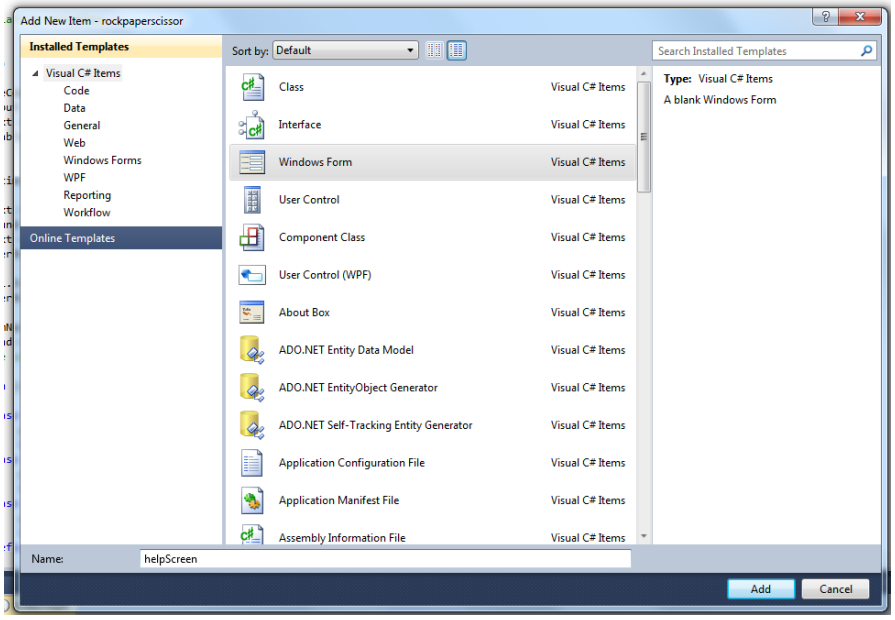
This is why it's always a good idea to create Help Window.

Now lets once again right click on the rockpaperscissors inside the solution explorer

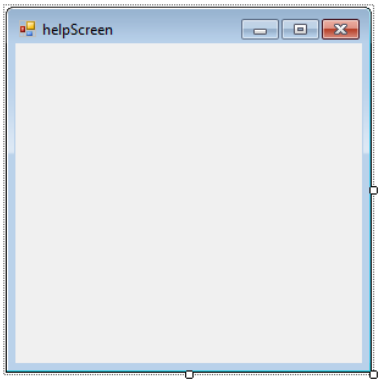


Add -> Windows Form

Name it helpScreen

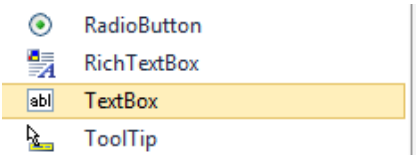


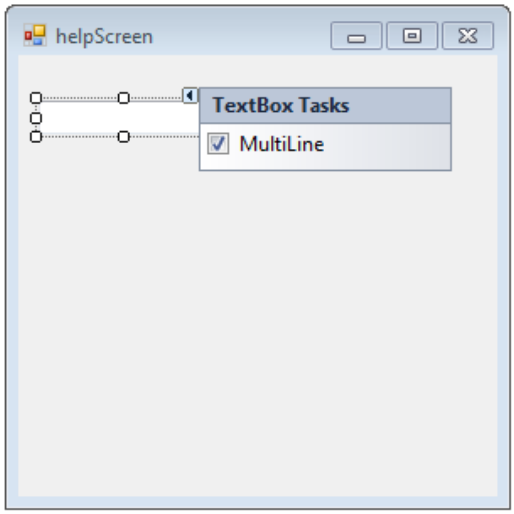
Click OK.



Now we can add any information which the user will find useful and will aid them in using this application / game.

Drag and drop a text box in this form



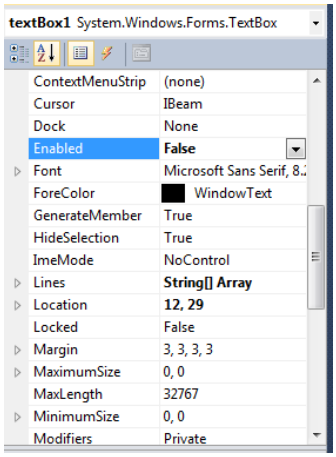


click on that small triangle on the right corner and select multi line.

Add the following sentences in the text box

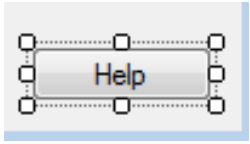
Welcome to the rock paper and scissor game create by (your name here) Enter your name in the beginning Select the option for rock, paper or scissor The countdown goes from 5 to 1 Once its completed the AI will show its choice to you Rules of the game are Rock beats scissor Paper covers rock Scissor cuts paper You have 3 rounds of game play Good Luck and Enjoy

Once you have entered the text inside the text box now look into the properties window for a option called enabled. Change it false. If it's set to true then the user can change the text or even delete them. Since its set to false it will be disabled they will only be able to read the whole text not edit it in any way.



Now let's do the following code for the help screen.

Double click on the help button



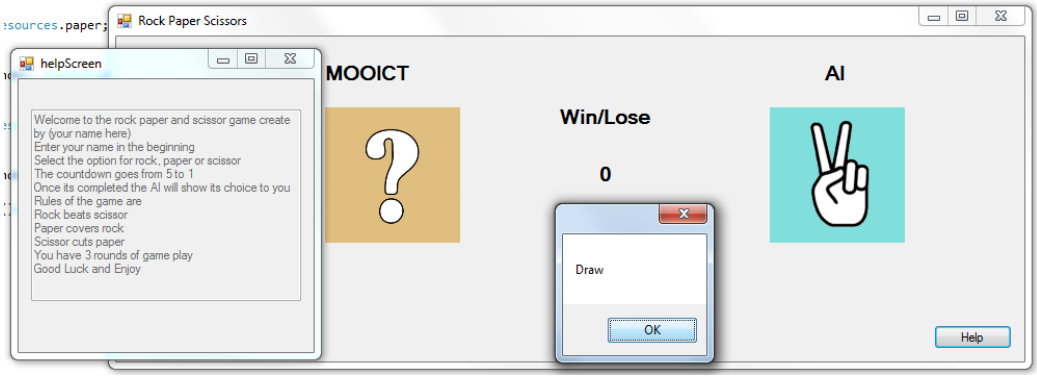

```
privatevoid button4_Click(object sender, EventArgs e)
        {
helpScreen help = newhelpScreen();
help.Show();
        }
```

Since we called the windows form helpScreenits classified as its own class so we can simply call the helpScreen directly. We are giving the helpScreen class a new instance of help and then we are linking it.

Since help is the new instance of the helpScreen class now we can access all of its properties through the code.

We are now able to call the help.Show() function and once this runs it will display the help form on screen.

Test it out.



Success

Well done following this tutorial thus far. Now you can add your own features to it. Good luck.

For the full code check the **page 2** of this tutorial post on mooict.com

Read Next: 1 2