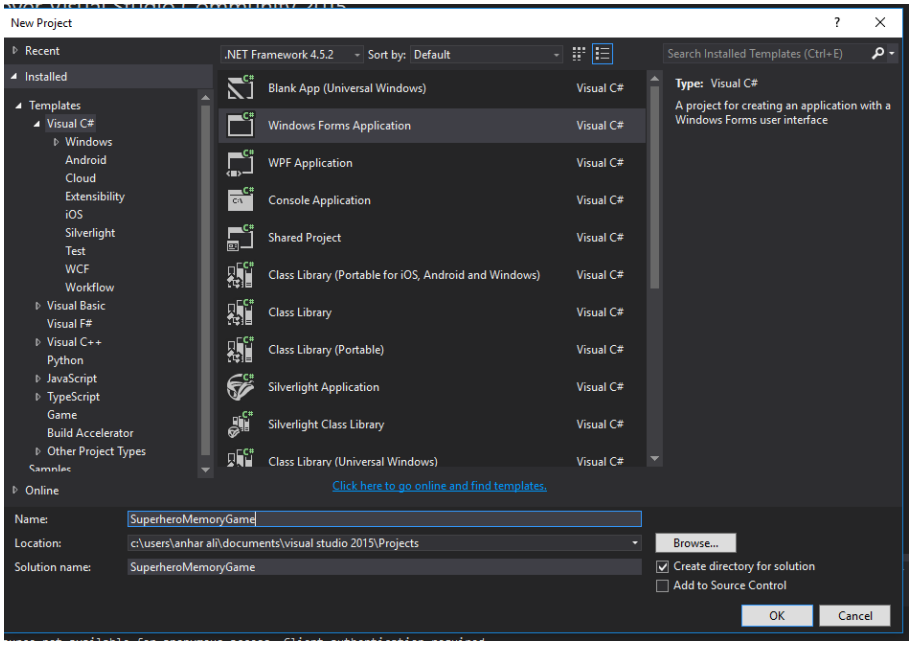
**C# Tutorial – Create a Superhero Memory Matching Game**

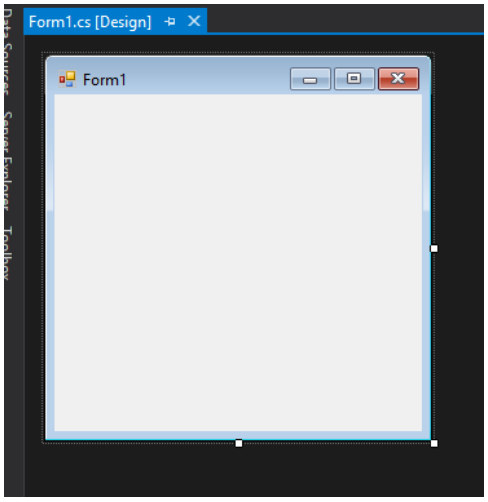
This tutorial will help you learn how to create a simple matching/memory game. We chose super hero images because they are the most recognisable and its fun. No copy right infringement intended for this tutorial, its purely for educational purposes.

With that all cleared up let's get learning.

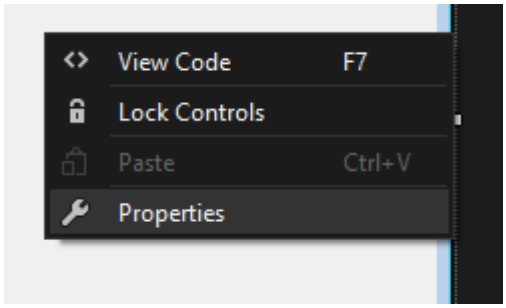
Start a new project in Visual Studio – Under C# chose the Windows Form Application.




Name this project **SuperheroMemoryGame** (no spaces). Once you click OK it will save it in the documents/visual studio 2015 folder, unless you have a different version then please double check it.



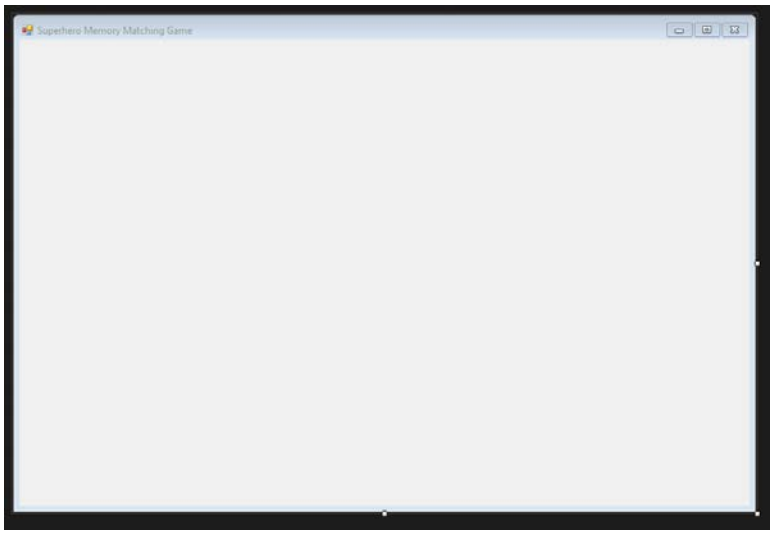

This is the empty form at the moment. We will be using the properties window A LOT so its best if you can open it now.




**More tutorials on** [www.mooict.com](http://www.mooict.com)

Right click on the form and click on properties. This will allow you to change various settings on that components. Get used to this, it has lots of options to make your project look and feel exactly like you want.

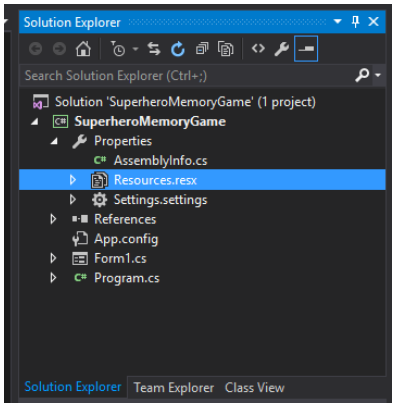

Find the size option in the properties panel and change the size to 913, 611



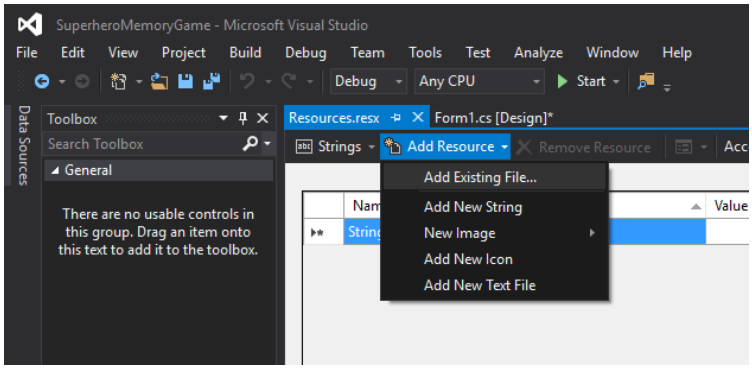

This is what the new size looks like.

Now we need to add the images we created for this game. It's a mixture of super heroes and a question mark image.

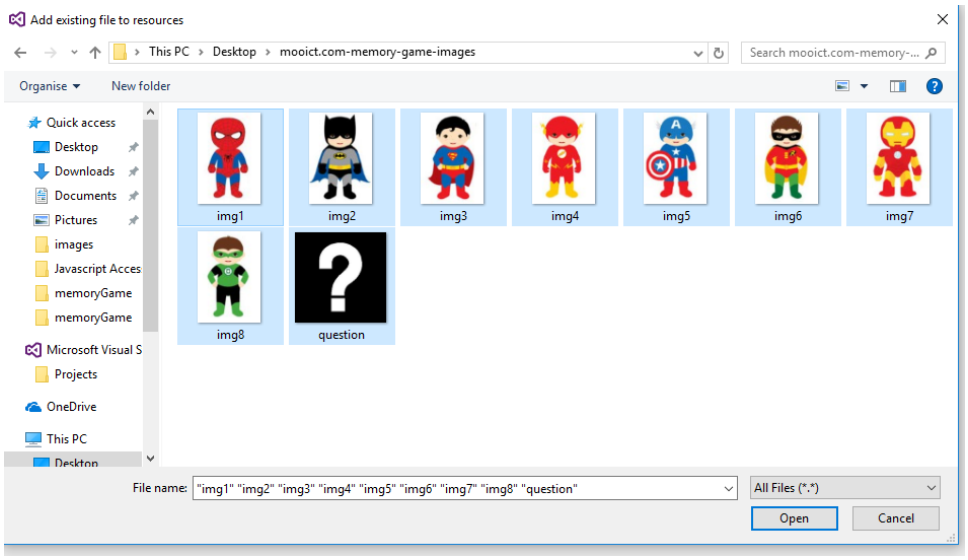
Go to the Solutions explorer on the right and click on the little arrow next to the properties option under the super hero memory game. Once you click on it will open several more options now double click on the **resources.resx**



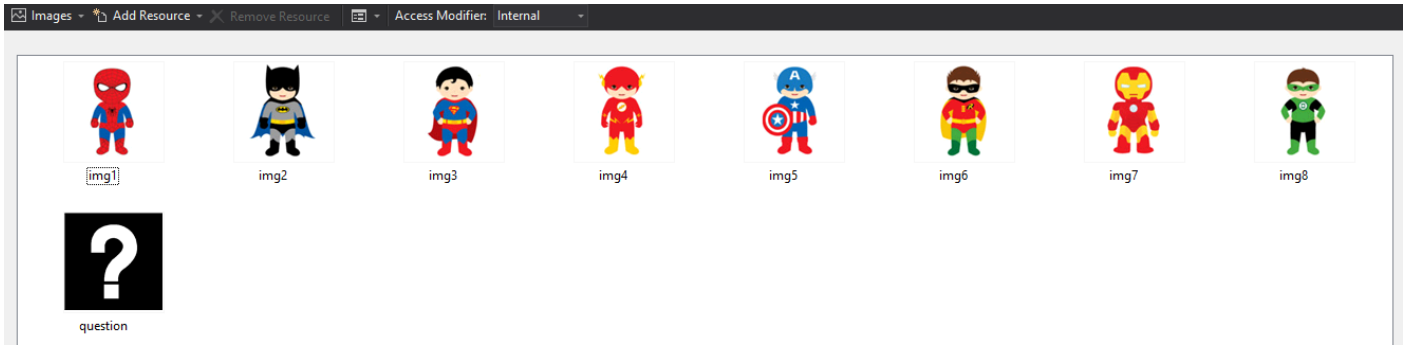

Double click on the **resources.resx**



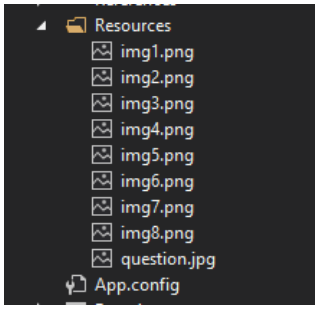

Click on the add resource drop down option and click on the add existing file.




Find the images you have extracted from the zip file, if you don't have the files yet download them from MOOICT tutorial page. There should be a link about with the images inside a ZIP file. Select all of the images and click on open.

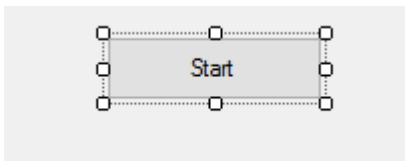



After you done that this is what the resource window looks like. Now remember to save this. CTRL+S. or file -> save all.



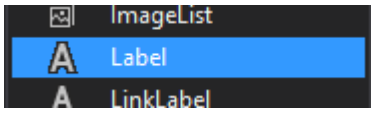

If you look at the solutions explorer now we gave a new folder called resources and it has all of the images we imported into this project earlier.


From the tool box find the button component and drag it down to the form.



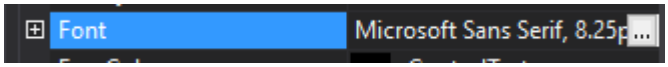

**More tutorials on** [www.mooict.com](http://www.mooict.com)

Go to the properties window and find the option called text for the button and change text to Start. This button will be used to start the game initially when it loads to screen.

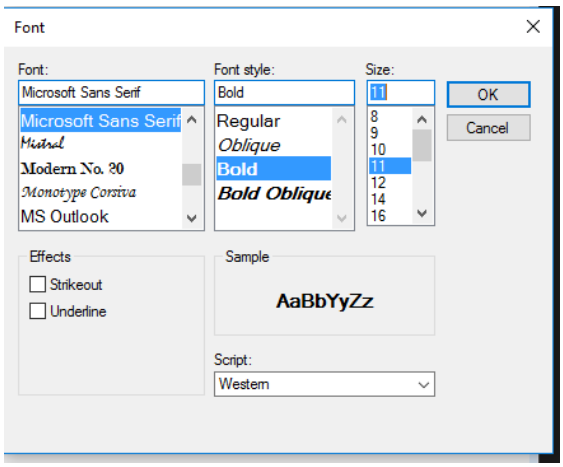



Now find a Label component in the tool box and drag it to the form.


Place the label under the button

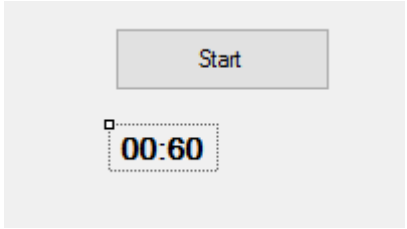



While the label is selected, go to the properties window and find the option called font – click on the … dotted button.

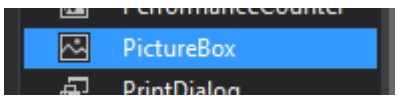



In this dialog box we can change the font, so we chose to change it to Bold and size to 11.

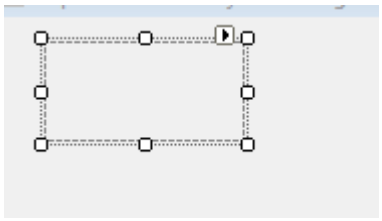

Change the text in the properties window to 00:60.



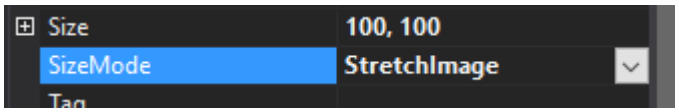

This is the final label.



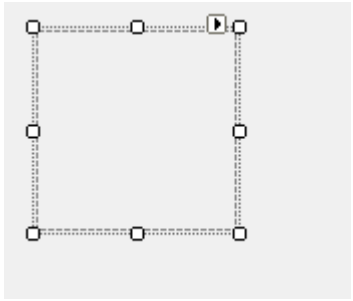

Now find the picture box component in the tool box, and drag it to the form.



This is the picture box on the form.

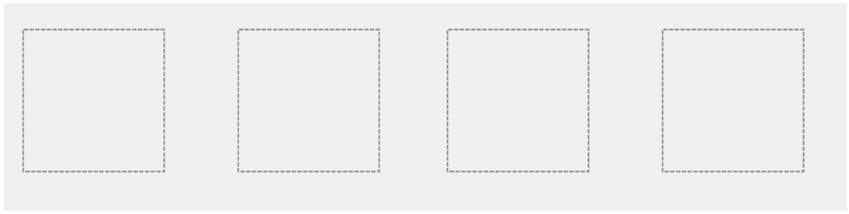



Change the following in the properties window for the picture box. Size to 100, 100 and size mode to stretch image.

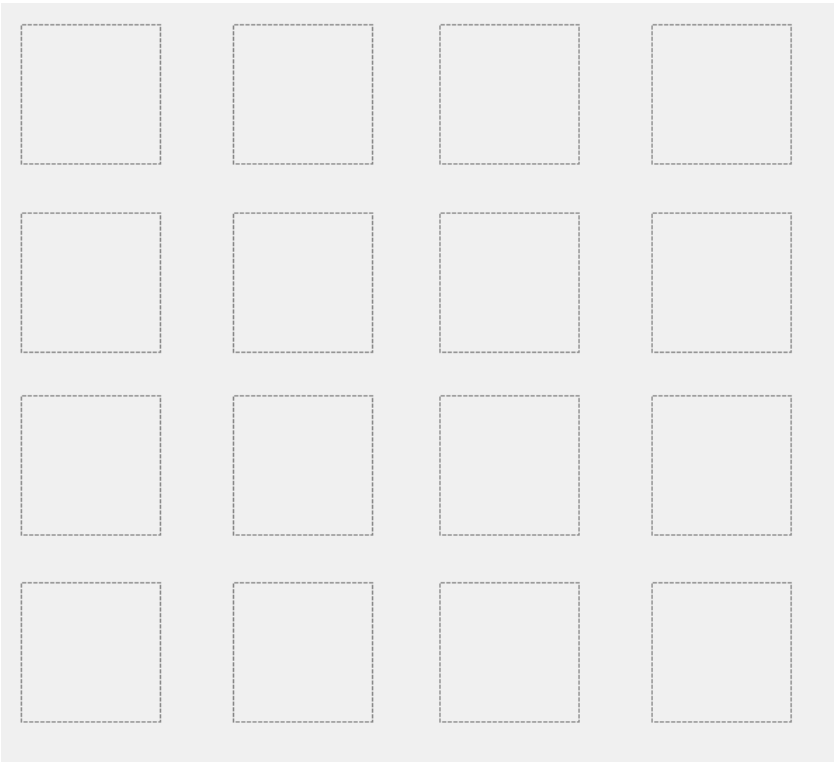


This is the final design on the picture box. Note – we are going to add the images we imported to the project earlier, we will be using C# code to dynamically import the images into these picture boxes.

We will need 16 picture boxes all together, so instead of creating them one by one we will copy and paste the one we created earlier in to 4.

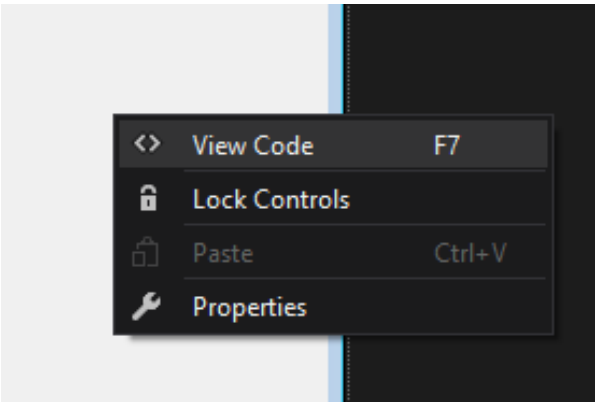


This is the view thus far.

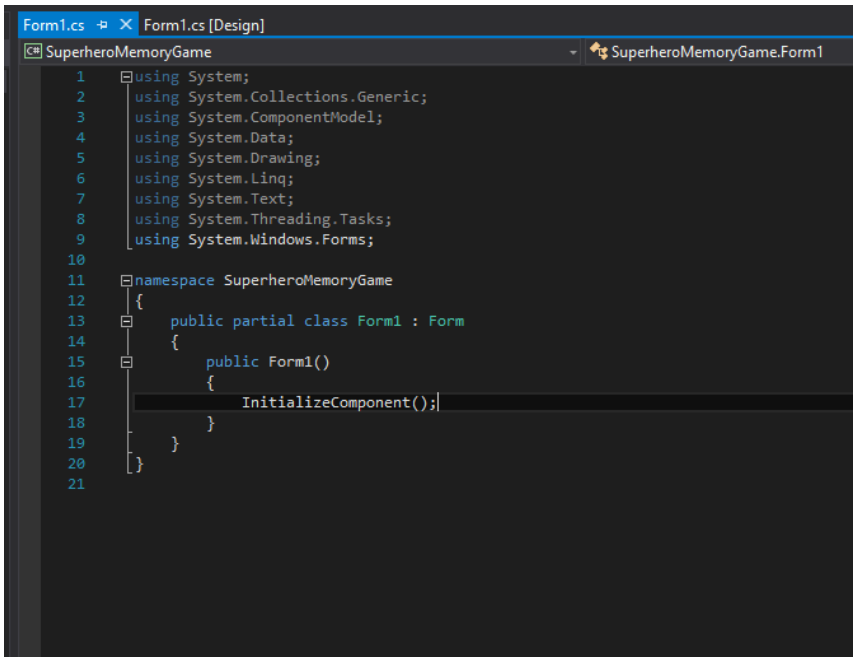


Now copy and paste the 4 picture boxes 3 times until you get a screen similar to this above.

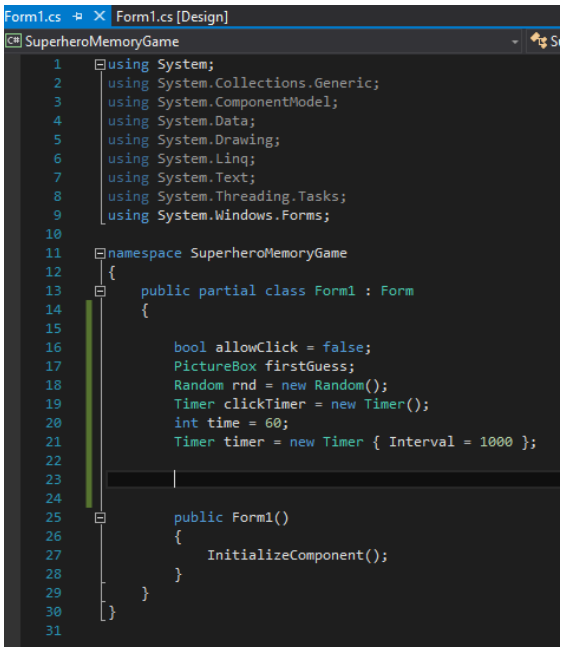
Now we are done with the GUI time to get coding.




Right click on the form and click on View Code.




This is the initial code view, nothing has been added yet. We are going to start by adding variables to this game.




Add the variables before the public Form1() line. These are global variables, since we will be using them in different functions its best to declare them at the top.

```
        bool allowClick = false;
        PictureBox firstGuess;
        Random rnd = new Random();
        Timer clickTimer = new Timer();
        int time = 60;
        Timer timer = new Timer { Interval = 1000 };
```

Bool allowClick = false; this is the Boolean called allowClick and its been set to false.

PictureBox fireGuess; We are creating an instance of a picture box object in a variable and naming it first guess.

Random rnd = new Random(); this line is creating a new instance of the Random number generator class.

Timer clickTimer = new Timer(); this line creating an instances of the timer class.

int time = 60; this is an integer which has a set value of 60. This will be used to calculate the countdown timer in the game.

Timer timer = new Timer {Interval = 1000} this is another timer we are adding to the game, this will have an Interval activated from the beginning which will run 1000 milliseconds or 1 second.

```
        private PictureBox[] pictureBoxes
        {
            get { return Controls.OfType<PictureBox>().ToArray(); }
        }
```

This is the PictureBox function, we are going to add all the pictureboxes to an Array. Enter this function right under the public Form1() function. Remember to start entering the function after the closing curly brackets }.

```
        private static IEnumerable<Image> images
        {
            get
            {
                return new Image[]
                {
                    Properties.Resources.img1,
                    Properties.Resources.img2,
                    Properties.Resources.img3,
                    Properties.Resources.img4,
                    Properties.Resources.img5,
                    Properties.Resources.img6,
                    Properties.Resources.img7,
                    Properties.Resources.img8
                };
            }
        }
```

Enter the function above as is. This functions purpose is to link the images which we imported to the resources earlier. Notice we are using a static IEnumerable <image> in the title of this function. What this means is we are going to create an array of Image class and we need to have access to those files. By using IEnumerable it makes it easy for us by creating an access to query those files.

```
        private void startGameTimer()
        {
            timer.Start();
            timer.Tick += delegate
             {
                 time--;
                 if (time < 0)
                 {
                     timer.Stop();
                     MessageBox.Show("Out of time");
                     ResetImages();
                 }

                 var ssTime = TimeSpan.FromSeconds(time);

                 label1.Text = "00: " + time.ToString();
             };
        }
```

This functions purpose is to start the timer and display the remaining time on the label we added earlier.

```
        private void ResetImages()
        {
            foreach (var pic in pictureBoxes)
            {
                pic.Tag = null;
                pic.Visible = true;
            }

            HideImages();
            setRandomImages();
            time = 60;
            timer.Start();
        }
```

This function will be resetting the picture boxes. This function will be used when we are allowing the user to play the game again after their first round.

```
        private void HideImages()
        {
            foreach (var pic in pictureBoxes)
            {
                pic.Image = Properties.Resources.question;
            }
        }
```

This function will run a foreach loop through the form looking for picture box components and it will mask them with the question mark image we imported earlier. Notice the propertie.Resources.question.

```
        private PictureBox getFreeSlot()
        {
            int num;

            do
            {
                num = rnd.Next(0, pictureBoxes.Count());
            }
            while (pictureBoxes[num].Tag != null);
            return pictureBoxes[num];
        }
```

This function has a data return type. Which means when this runs it will return a value back to the program. In this function we are first declaring an integer num, then we are going to run a do LOOP which will loop through all of the picture boxes randomly. While loop will check which picture boxes tags are = null or EMPTY and then we will return those back to the program. By doing this process we can randomly select a pair of picture boxes. This process will be useful for the function where we set the random images.

```
        private void setRandomImages()
        {
            foreach (var image in images)
            {
                getFreeSlot().Tag = image;
                getFreeSlot().Tag = image;
            }
        }
```

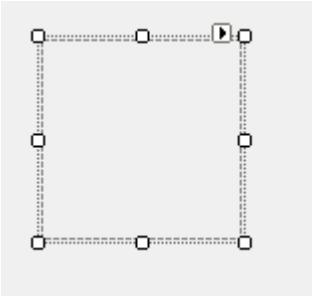
In this function above we are running another loop, this time instead of running a DO WHILE loop we are doing a for each loop, this loop will look for images and try to find a pair of slots where both can be tagged with the same name, thus giving us a change to match the images with each other. See how we are using getFreeSlot() function, because the data type of the function is set to Picture Box therefore we can invoke the object property of TAG from there.

Next is the click timer function.

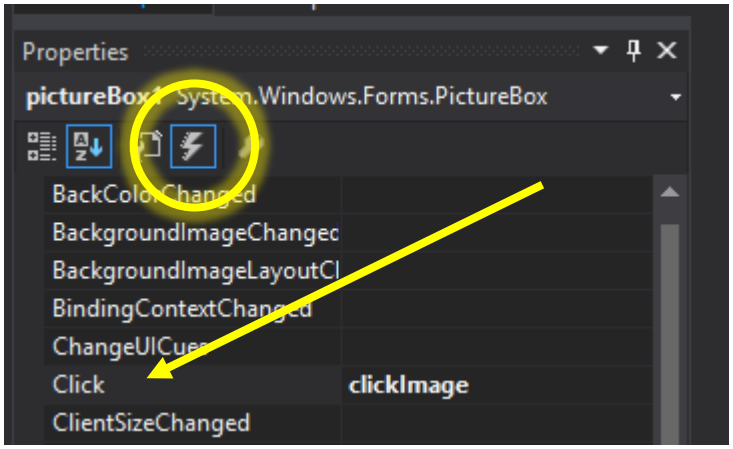
```
        private void CLICKTIMER_TICK(object sender, EventArgs e)
        {

            HideImages();

            allowClick = true;
            clickTimer.Stop();
        }
```

The above function is simple, this function will first hide all of the images by using the HideImages() function [discussed a little later], will change the allow click Boolean to true and then stop the timer itself. This timer only has these instructions to run so it can stop when its successful.

Now we need to link an event to the picture boxes. Let's start with one and then we can add it to the rest once you know how to do it.



Click on the picture box, while its active look into the properties window.




Click on that lightning bolt icon, this icon presents the events catalogue for visual studio components. You will find all sorts of them here, if you followed the tutorials from MOOICT.COM before then you know when have gone in depth with events. For now, we are interested in the click event, find the click event option while picture box is selected and type clickImage (no spaces between them).

Press enter. This will automatically take you to the code screen, you will find the empty clickImage function. Enter the following code highlighted in blue below.

```
        private void clickImage(object sender, EventArgs e)
        {
            if (!allowClick) return;

            var pic = (PictureBox)sender;

            if (firstGuess == null)
            {
                firstGuess = pic;
                pic.Image = (Image)pic.Tag;
```

```
                return;

            }

            pic.Image = (Image)pic.Tag;

            if (pic.Image == firstGuess.Image && pic != firstGuess)

            {
                pic.Visible = firstGuess.Visible = false;
                {
                    firstGuess = pic;

                }
                HideImages();
            }
            else
            {
                allowClick = false;
                clickTimer.Start();
            }

            firstGuess = null;
            if (pictureBoxes.Any(p => p.Visible)) return;
            MessageBox.Show("You Win Now Try Again");
            ResetImages();
        }
```

First we are running a if statement, is isClicked is false then return back to the program without doing anything else. Notice it has **!** symbol is front of it that means if allowClick is false (!allowClick).

Var pic = (picturebox)sender; in this line we creating a local variable, which will only be used inside this function called pic. This variable will identify which picture box was clicked or where this event came from.

```
if (firstGuess == null)
            {
                firstGuess = pic;

                pic.Image = (Image)pic.Tag;

                return;

            }
```

If the first guess is bull or empty, then we are going to allow the pic variable to become our first guess, since the pic variable is a type of picture box it has the same properties as one, therefore we can use the pic.Image property to set any images to it. In the later line we are setting an image to the pic variable using (Image)pic.Tag value. Finally, we are returning to the program.

```
pic.Image = (Image)pic.Tag;
```

when the images are found, it will set the appropriate tag to the picture box.

```
firstGuess = null;
```

outside the is statement we are doing the same, this way we can reduce the number of errors we encounter. If a null value or empty value is detecting usually the program will throw some error at us but by making the program look for the null and give it a counter path to take we are reducing the errors.

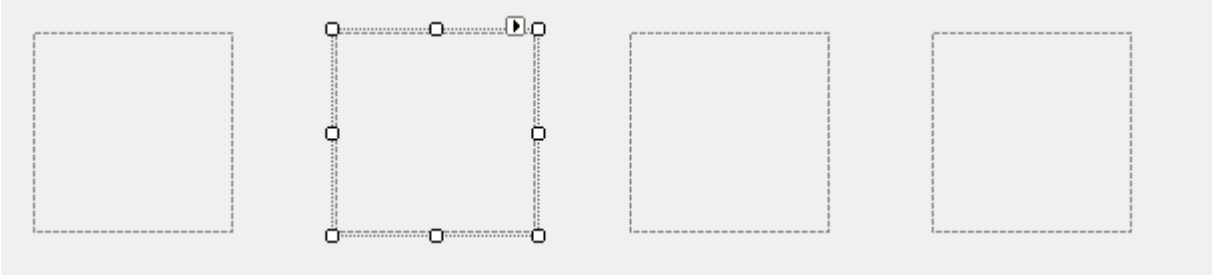
```
if (pictureBoxes.Any(p => p.Visible)) return;
```

This line is checking if there any visible picture boxes left on the screen. If there are we continue to play the game if there are not then we run the line below it.

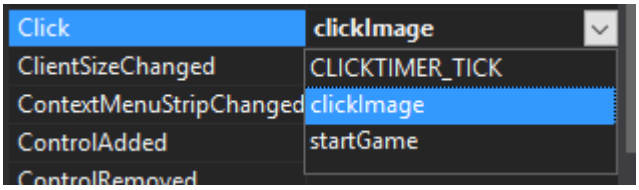
```
MessageBox.Show("You Win Now Try Again");
```

If all of the picture boxes are matched and done then we run this message box to show the player has won and run the Set Images below. This will restart the game for the player.

```
ResetImages();
```

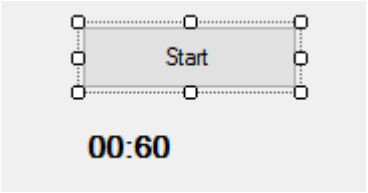


Now click on the other picture boxes and set the click event to click Image event we have created. Make sure to include all of the picture boxes because they all need to have the same event running and checking whether they have matched.

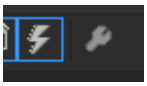



Go to the events properties you will be able to see the click Image function from the drop-down menu. Click on the single picture boxes – look at the event properties window – find click – from the drop down chose click Image function. – Done.

Finally let's add the start button functions.




Click on the start button



Go to the events properties window, find the click event – type **startGame** and press enter.

```
        private void startGame(object sender, EventArgs e)
        {
            allowClick = true;
            setRandomImages();
            HideImages();
            startGameTimer();
            clickTimer.Interval = 1000;
            clickTimer.Tick += CLICKTIMER_TICK;
            button1.Enabled = false;
```

```
        }
```

Inside the start game function add the highlighted code.

First, we are setting the allow click Boolean to true. Run the set random images function, also run hide images function which will fill the boxes with question marks, after that set the click timer interval to 1000 millisecond or 1 second, then we are linking the click timer to a click timer tick function and finally disable the start button.

*Disclaimer –*

*This tutorial was inspired by the matching game tutorial on you tube which is in Russian language. We have translated the intended purpose as much as possible in English and hope you all enjoyed it.*

*Original you tube video by - **Cyber Code** if you like their content give them a sub and video link is - https://www.youtube.com/watch?v=n69_q2MbjLo*

Full Code of 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 memoryGame
{
    public partial class Form1 : Form
    {

        bool allowClick = false;
        PictureBox firstGuess;
        Random rnd = new Random();
        Timer clickTimer = new Timer();
        int time = 60;
        Timer timer = new Timer { Interval = 1000};


        public Form1()
        {
            InitializeComponent();
        }

        private PictureBox[] pictureBoxes
        {
            get { return Controls.OfType<PictureBox>().ToArray(); }
        }

        private static IEnumerable<Image> images
        {
            get
            {
                return new Image[]
                {
                    Properties.Resources.img1,
                    Properties.Resources.img2,
```

```
                Properties.Resources.img3,
                Properties.Resources.img4,
                Properties.Resources.img5,
                Properties.Resources.img6,
                Properties.Resources.img7,
                Properties.Resources.img8
            };
        }
    }

    private void startGameTimer()
    {
        timer.Start();
        timer.Tick += delegate
         {
             time--;
             if (time < 0)
             {
                 timer.Stop();
                 MessageBox.Show("Out of time");
                 ResetImages();
             }

             var ssTime = TimeSpan.FromSeconds(time);

             lblTime.Text = "00: " + time.ToString();
         };
    }

    private void ResetImages()
    {
        foreach (var pic in pictureBoxes)
        {
            pic.Tag = null;
            pic.Visible = true;
        }

        HideImages();
        setRandomImages();
        time = 60;
        timer.Start();
    }

    private void HideImages()
    {
        foreach (var pic in pictureBoxes)
        {
            pic.Image = Properties.Resources.question;
        }
    }

    private PictureBox getFreeSlot()
    {
        int num;

        do
        {
            num = rnd.Next(0, pictureBoxes.Count());
        }
        while (pictureBoxes[num].Tag != null);
        return pictureBoxes[num];
    }

    private void setRandomImages()
    {
        foreach (var image in images)
        {
```

```
                getFreeSlot().Tag = image;
                getFreeSlot().Tag = image;
            }
        }

        private void CLICKTIMER_TICK(object sender, EventArgs e)
        {

            HideImages();

            allowClick = true;
            clickTimer.Stop();
       }

        private void clickImage(object sender, EventArgs e)
        {
            if (!allowClick) return;

            var pic = (PictureBox)sender;

            if (firstGuess == null)
            {
                firstGuess = pic;
                pic.Image = (Image)pic.Tag;
                return;

            }

            pic.Image = (Image)pic.Tag;

            if (pic.Image == firstGuess.Image && pic != firstGuess)

            {
                pic.Visible = firstGuess.Visible = false;
                {
                    firstGuess = pic;

                }
                HideImages();
            }
            else
            {
                allowClick = false;
                clickTimer.Start();
            }

            firstGuess = null;
            if (pictureBoxes.Any(p => p.Visible)) return;
            MessageBox.Show("You Win Now Try Again");
            ResetImages();
        }

        private void startGame(object sender, EventArgs e)
        {
            allowClick = true;
            setRandomImages();
            HideImages();
            startGameTimer();
            clickTimer.Interval = 1000;
            clickTimer.Tick += CLICKTIMER_TICK;
            button1.Enabled = false;
        }
    }
}
```