C# Animating Multiple Objects on Windows Form

In this tutorial we will learn how to animate multiple objects on a Windows Form application in C#. We are going to use the Microsoft Visual Studio 2010 version; you can use any other to achieve the same result.

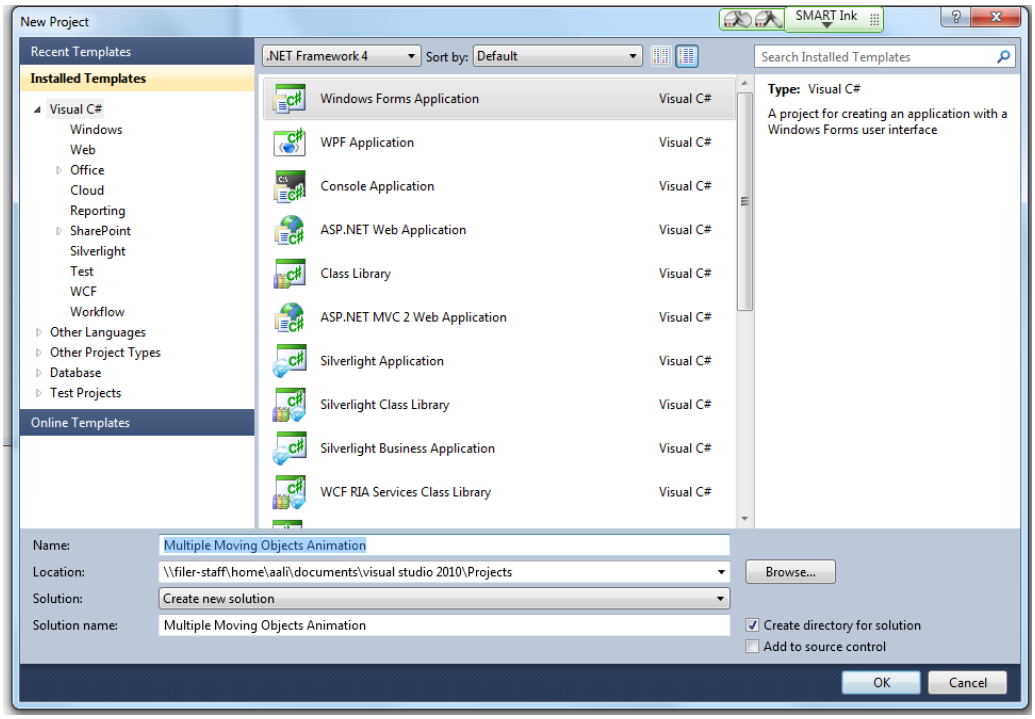
Lesson objectives –

- To understand the timer function in C#
- Reduce and increasing the timer functions events
- Using the tool bar effectively
- Animating multiple objects at different speeds in windows form application using C#

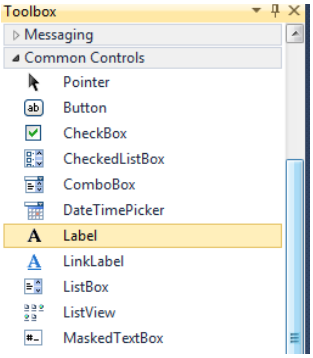
For this example we will animate 3 different labels on the screen at different speeds, you can use the same code to animate picture boxes, movie clips or any other objects on screen.

This process of animation can be used to create games where you can have one main timer which controls the animation of all three labels on the screen.

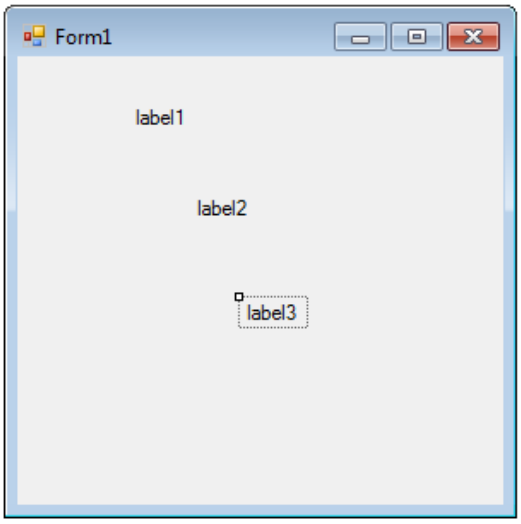
Start Visual studio, under the C# language choose Windows Form application.



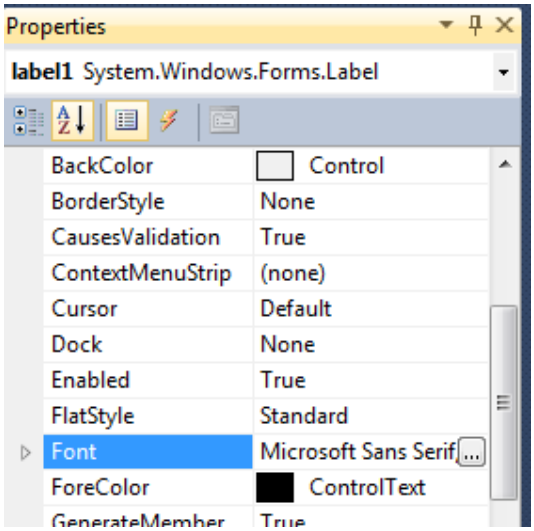
Name the project Multiple Moving Object Animation



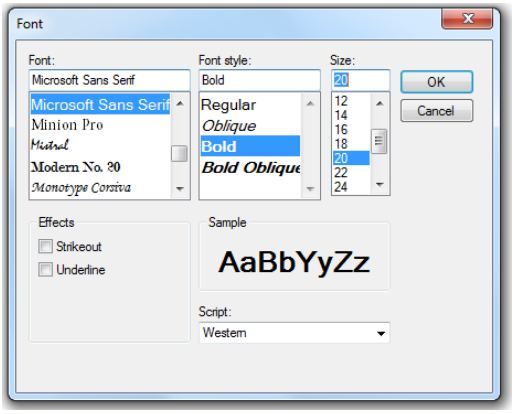
We will need 3 labels on the screen, for this example we won't be giving the labels are names.



Here are the 3 labels on the screen. To make things more visible we need to change the font settings of the labels.

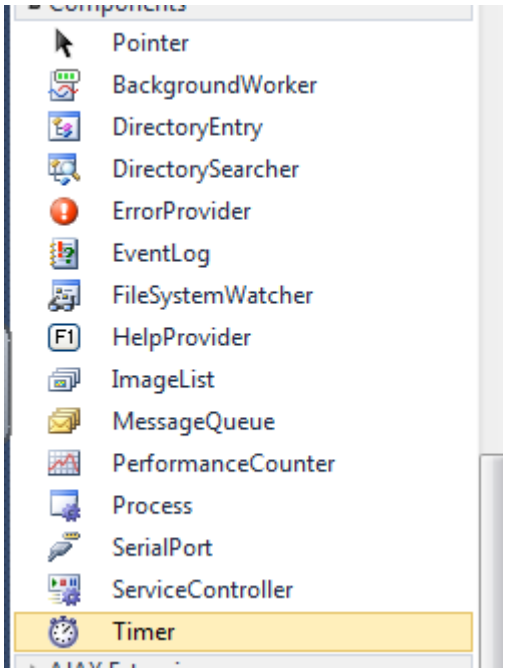


Click on one of the labels and you will see the font option in the properties window. Click on the … and it will open the window below.



Here we will change the settings to BOLD and font size 20. CLICK OK when done.

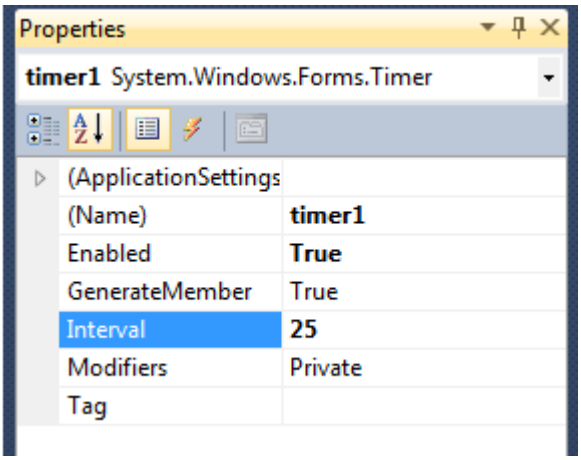
Here is the final view of the form with all three labels changed to the desired font properties.



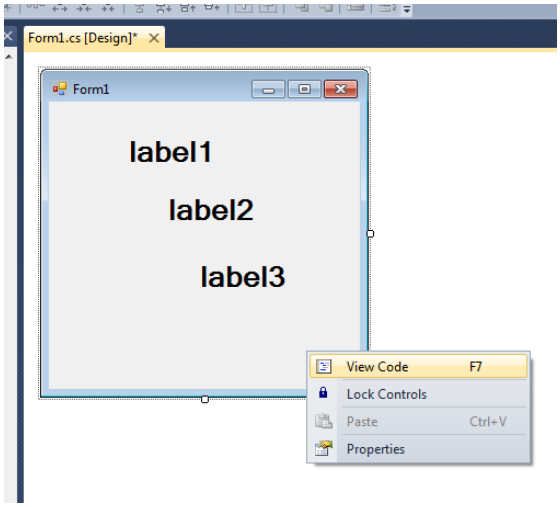
Drag and drop a TIMER object from the tool box to the form.

Time object doesn't appear on the form itself, it will appear bottom of the form on a separate tool bar.

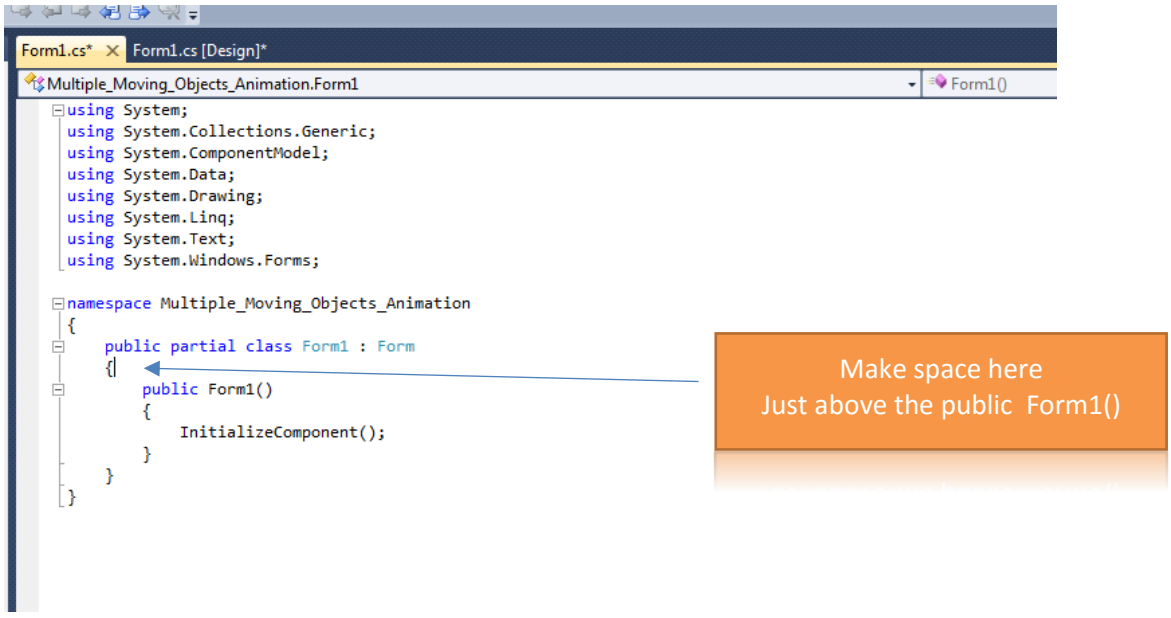
Click on the timer1 object and we will need to make the following changes to the properties window.



Change the ENABLED option to TRUE and change the INTERVAL to 25. This means when the form will run the time will run from start. The interval will fire up the function for us every 25 milliseconds.



Right click on the form and click on VIEW CODE or press F7

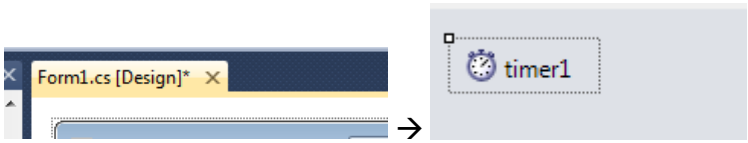


Above the public form1() line lets add some space to insert the global variables.

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

namespace Multiple_Moving_Objects_Animation
{
    public partial class Form1 : Form
    {

        int L1x = 5;
        int L1y = 5;



        public Form1()
        {
            InitializeComponent();
        }


    }
}
```

Since L1x and L1y are not defined INSIDE any individual function they are **global variable**s which means these two variables are now accessible in any function throughout the program.

Above we have added 2 integer variables called L1x and L1y and we have given them a value of 5. This will be speed for the X and Y axis for label 1. Let's go back to the form design view and double click on the TIMER object.



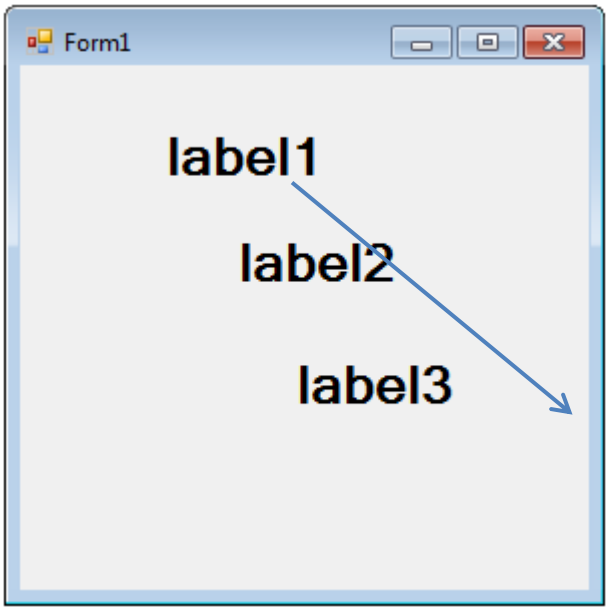
← Double click on this.

```
private void timer1_Tick(object sender, EventArgs e)
{

}
```

Once double clicked on the object, this line of code will be added to the code. This is the event that will fire up every 25 milliseconds. All of the logic for the animation will be inside this.

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            label1.Left += L1x; //this will move it horizontally
            label1.Top += L1y; // this will move it vertically


        }
```

Click on the debug button – the green play button on the tool bar.



Now you can see that the label 1 will go diagonally and disappear. We don't want it to go off screen but we want to it to bounce from border to border. The lines added below will allow us to achieve that effect.

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            //label 1 movement starts
            label1.Left -= L1x; //this will move it horizontally
            label1.Top -= L1y; // this will move it vertically

            if (label1.Left + label1.Width > ClientSize.Width || label1.Left < 0)
            {
                L1x = -L1x; // this will bounce the object from the left or right border
            }

            if (label1.Top + label1.Height > ClientSize.Height || label1.Top < 0)
            {
                L1y = -L1y; // this will bounce the object from top and bottom border
            }
        }
```

This is a comment line usually starts with //

This part means if the label has gone out of the boundaries from RIGHT

This symbol || means OR

This part will check if the label has gone out of boundaries from the LEFT

This line above has two different if statements in it. Now focus inside the if statement which is stating clearly that if label 1 left and label 1 width added together is greater than the border width left or right then we are going to change the L1X variable value from positive to negative. This will do the opposite effect on the label and change directions from right to left. The if statement will read if the label has left the right OR the left then does the following inside the curly brackets.

The theory of IF statement goes like this -

IF (THIS HAPPENS INSIDE THIS BRACKET – MULTIPLE CONDITIONS CAN BE APPLIED IN THERE TOO)

{ DO THESE ACTIONS FROM THESE CURLY BRACKETS}

Second if statement is going to stop the label from crossing the top and bottom border. So if the label 1 top and label 1 height is added together and is great than the border height then we are going to change the L1Y from positive to negative which will give us the bounce effect.

If you run the program now it will simply bounce around the screen.

Our job is not done yet we still have two more objects to animate.

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Multiple_Moving_Objects_Animation
{
    public partial class Form1 : Form
    {

        int L1x = 5;
        int L1y = 5;

        int L2x = 5;
        int L2y = 5;
```

Above we added second pair of variables for the Label 2. Since L1X and L1Y are being used for label 1, we cannot reuse them for label 2.

```
        private void timer1_Tick(object sender, EventArgs e)
        {
            //label 1 movement starts
            label1.Left -= L1x; //this will move it horizontally
            label1.Top -= L1y; // this will move it vertically

            if (label1.Left + label1.Width > ClientSize.Width || label1.Left < 0)
            {
                L1x = -L1x; // this will bounce the object from the left or right border
            }

            if (label1.Top + label1.Height > ClientSize.Height || label1.Top < 0)
            {
                L1y = -L1y; // this will bounce the object from top and bottom border
            }
            // label1 movement ends

            // label 2 move starts here

            label2.Left += L2x; //this will move it horizontally
            label2.Top += L2y; // this will move it vertically

            if (label2.Left + label2.Width > ClientSize.Width || label2.Left < 0)
            {
                L2x = -L2x; // this will bounce the object from the left or right border
            }

            if (label2.Top + label2.Height > ClientSize.Height || label2.Top < 0)
            {
                L2y = -L2y; // this will bounce the object from top and bottom border
            }

            // label 2 movement ends here
        }
```

The highlighted code is for label 2. You can see the similarities with the label 1 functions and if statements. We are essentially doing the same with this one as we did with the one before. We are checking the whether the label 2 is leaving the width of the borders left or right and if it is we will change the L2X to a negative value so it bounces to the other direction. Then we will check whether its crossing over from the top or bottom and we will make it bounce on the other direction aswell.

Task – Now you can make the LABEL 3 animate on your own. Check to see how the other two were done and do it yourself.

All the best. Moo Out