

Tutorial – Windows Forms and Basic Controls

In this tutorial we are going to be creating a simple program that could form the foundations of a waypoint editing program.

By the end of this tutorial you will have a Windows Forms Application that will allow the user to click on a picture and display the x and y coordinates of the user's click in a text box.

Subsequent tutorials will cover more advanced concepts that you can add to this program.

Although we won't be building a complete waypoint editor in these tutorials, by the end you should understand all the concepts necessary for you to complete the assignment for this subject.

Be sure to also complete the exercises at the end of each tutorial, as each exercise is designed to get you closer to finishing your assignment.

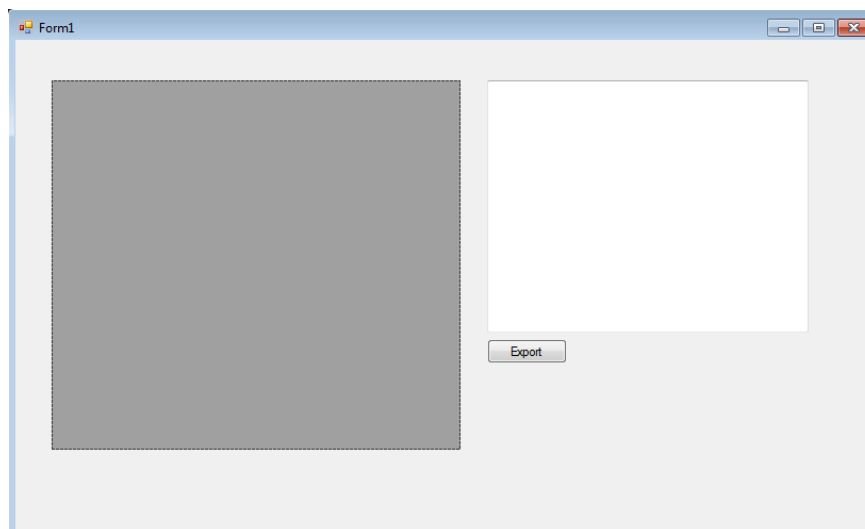
Creating a New Windows Forms Application:

1. Open Visual Studio 2013 and create a new C# Windows Forms Application project in a new solution
2. If the Forms Designer does not open, double-click the Form1.cs file in the Solution Explorer
3. Open the toolbox and start placing controls onto your form.

There are three controls we want to add.



- a PictureBox,
 - a TextBox, and
 - a Button
4. Open the Properties window if it is not already visible and make the following changes to the components you added:
 - a. Set the **BackColor** of the PictureBox to something other than the default. (You don't need to set a picture, but you can if you want. In a later tutorial we'll replace the PictureBox with our own custom control)
 - b. Change the **Name** of the TextBox to *textOutput*, and set the **Multiline** property to *True*.
 - c. Change the **Name** of the Button to *buttonExport*, and set the **Text** to *Export*

After making these changes and arranging the controls on the form to your liking, your form should look something like this:

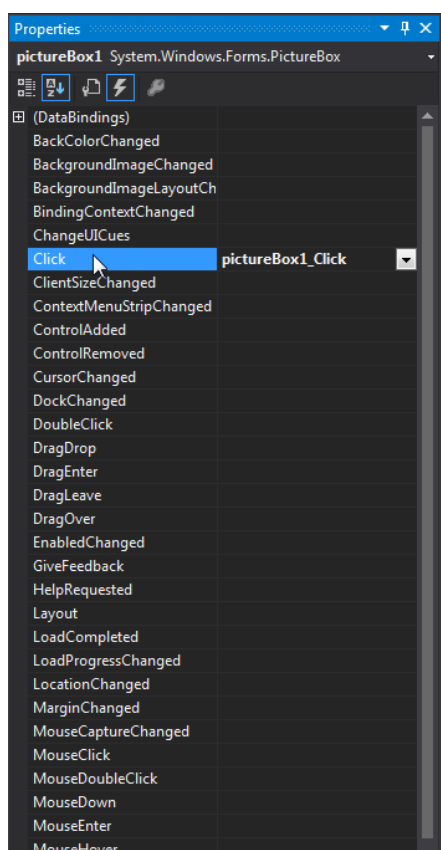


Adding Events:

1. Select the PictureBox so that the control's properties are visible.

In the Properties window, select the Lightning Bolt icon . This will open the events for the control. (The Page and Wrench icon  displays the control's properties)

With the events visible, double-click on the **Click** event entry. A new (coded) event will be created for you and if you return to the Properties window the **Click** entry will have updated with the name of the event handler.



2. Enter the following code for the PictureBox's Click event handler (in Form1.cs):

```
private void pictureBox1_Click(object sender, EventArgs e)
{
    if(e.GetType() == typeof(MouseEventArgs))
    {
        MouseEventArgs me = e as MouseEventArgs;
        textOutput.Text = me.Location.ToString();
    }
}
```

When the user clicks on the PictureBox, the mouse coordinates will be written to the TextBox

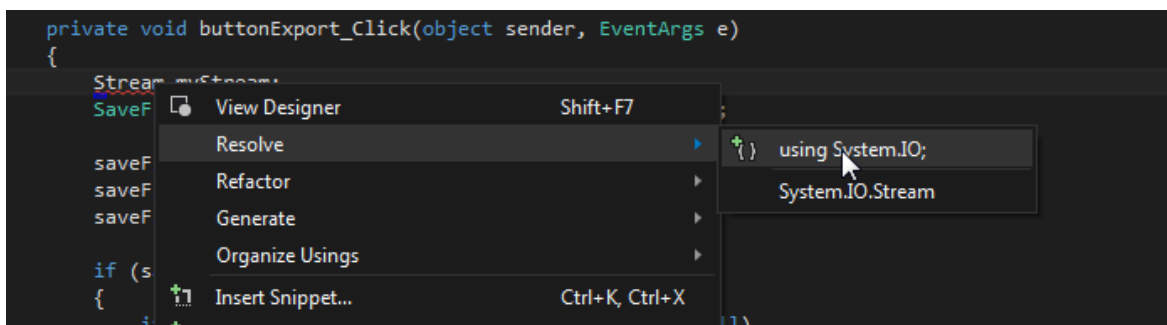
3. Go back to the Forms Designer and open the Events for the Button control. Add an event handler for the Button's **Click** event.
4. Enter the following code for the Button's click event handler:

```
private void buttonExport_Click(object sender, EventArgs e)
{
    Stream myStream;
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();

    saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
    saveFileDialog1.FilterIndex = 2;
    saveFileDialog1.RestoreDirectory = true;

    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        if ((myStream = saveFileDialog1.OpenFile()) != null)
        {
            // Code to write the stream goes here.
            myStream.Close();
        }
    }
}
```

Be sure to resolve any dependencies.



This code will display the Save File dialog when the button is pressed.

It will create a new output stream which you can write to. Anything you want to save to the output text file should be written to this stream.

Run your program and experiment with clicking on the PictureBox to see what output is written to the TextBox.

Exercises:

1. Complete the processing for the Export button by writing the text in the TextBox to the output file.
2. Currently each time you click on the PictureBox the text in the TextBox is reset. Modify the program so that each point you add is appended to the end of the text in the TextBox.
3. Add a new button to the program that will clear the text in the TextBox.
4. Create a new class to store the coordinates of the mouse clicks. Each time the user clicks on the PictureBox, take the coordinate data and create a new object. Store the objects in a collection (for example, a List<>). Ensure that all the elements in this list are shown as text in the TextBox.