

C#

C SHARP is Microsoft's development of C++, and incorporates object oriented methods more so than does C++. It also incorporates more consistent packaging of what would be extra downloads in other systems. It is complete with an IDE, and the IDE does much of the syntax checking. For documentation, what works well is an internet search on "C# FUNCTIONS TEXT FLOAT TO INTEGER" or some such question. Usually the first few choices will provide the answer. The Visual Express packages were language specific whereas the Visual Studio packages which replaced them are computer specific while covering several languages all in the one package.

C# Overview

[http://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)#Implementations](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)#Implementations)

A free IDE ~ not needed as the Visual Studio package has an excellent IDE

<http://en.wikipedia.org/wiki/SharpDevelop>

Microsoft resources: <http://msdn.microsoft.com/en-us/vstudio/hh341490.aspx>

then you must register. This involved an email address and a password that you select. Then it sends a verification email with a link you click on to confirm. Do not be surprised if the link takes you to a page-not-found. I had to try several times.

For Windows 8 use the following, the other choice demands Windows 8.1 and it means it.



Express 2013 for Windows Desktop

Visual Studio Express for Windows Desktop lets you take full advantage of Windows with XAML designers, a productive IDE, and a variety of programming languages including C#, Visual Basic, and C++. Choose between Windows Presentation Foundation (WPF), Windows Forms, and Win32, to target the Windows desktop with the right technology for your application and your skills.

[Download](#)

System Requirements:

Supported operating systems

- Windows 7 SP1 (x86 and x64)
- Windows 8 (x86 and x64)
- Windows 8.1 (x86 and x64)
- Windows Server 2008 R2 SP1 (x64)
- Windows Server 2012 (x64)
- Windows Server 2012 R2 (x64)

Required components

- Internet Explorer 10

Hardware requirements

- 1.6 GHz or faster processor
- 1 GB of RAM (1.5 GB if running on a virtual machine)
- 5 GB of available hard disk space
- 5400 RPM hard drive

The Microsoft system works first time, and it does generate an exe file and that runs also. This is an improvement over the early editions of the Visual Express applications.

The system is downloaded and installed,, it wants your email and the password you chose for Visual Studio, expect to enter it several times. By the way, while the IDE is excellent at identifying source code problems, it is not overburdened with help, so, check the **NOTE:** sections of this article.

Download language
English

Installation options

- Microsoft Visual Studio Express 2013 for Windows Desktop - English
[Install now](#)
- Microsoft Visual Studio Express 2013 for Windows Desktop - English
DVD5 ISO image (SHA-1: E61419E51F42254EE07DECF628B85C9861286250)

Microsoft Visual Studio Express 2013 for Windows Desktop Language Pack - Český
[Download now](#)

Visual Studio Team Foundation Server Express 2013

Visual Studio Express 2012 for Windows Phone

then

Visual Studio

Express 2013 for Windows Desktop

Acquiring:
Microsoft Visual C++ 2013 x64 Debug Runtime - 12.0.21005

Applying:
Microsoft .NET Framework 4.5.1

Cancel

After download and installation, a computer restart is required. Then locate the system in the ALL PROGRAMS list as VISUAL STUDIO 2013 and then VS EXPRESS 2013 FOR DESKTOP, and right click and create a desk top entry. Once done, click on the shortcut on the desktop when it will want you to sign in, use the same userid and password as before.

Microsoft account

Sign in

Microsoft account

Password

Sign in

Don't have a Microsoft account? [Sign up](#)

[Privacy](#) | [Terms](#)

© 2014 Microsoft

Start Page - Microsoft Visual Studio Express 2013 for Windows Desktop

FILE EDIT VIEW DEBUG TEAM TOOLS TEST WINDOW HELP

Start Page

Express 2013 for Windows Desktop

Start

[New Project...](#)

[Open Project...](#)

[Open from Source Control...](#)

Recent

Discover what's new in Express 2013 for Windows Desktop

You can find information about new features and enhancements in Express 2013 for Windows Desktop by reviewing the following sections.

[Learn about new features in Express 2013 for Windows Desktop](#)
See what's new in .NET Framework 4.5.1
[Explore what's new in Team Foundation Service](#)

[Relocate the What's New information](#)

What's new in Windows Desktop

- [Visual C#](#)
- [Visual Basic](#)
- [Visual C++](#)
- [Windows Presentation Foundation \(WPF\)](#)
- [Windows Forms](#)

Getting Started

- [Build Win32/Visual C++ applications](#)
- [Build WPF applications](#)
- [Build Windows Form applications](#)
- [Discover extensions, add-ons and samples](#)

Announcements

[Visual Studio 2013.1 \(Update 1\) is available](#)
Sunday, December 22, 2013
[Read Brian Harry's note on the release of the first Visual Studio 2013 Update.](#)

[Visual Studio 2013.1 RC is Available](#)
Wednesday, December 11, 2013
[Read Brian Harry's note on the first Visual Studio 2013 Update.](#)

[Rename Refactor Helper for Visual Studio 2013](#)
Tuesday, December 3, 2013
[Check out this new DevLabs prototype, a "rename" refactoring tool for Visual C++.](#)

[MVPs Explore Visual Studio 2013](#)
Tuesday, December 3, 2013
[Read MVP posts diving into the features of Visual Studio 2013.](#)

Ready

At which point you may get started.

NOTE: A quick note, file names this system generates can be long. Where you can, make them meaningful and short. This is because when you copy them, the complete path and file name character count may exceed the file systems limits. This is a file and folder name issue, nothing to do with the size of the actual files themselves.

THE SIMPLEST KIND OF APPLICATION ~ CONSOLE APPLICATION TYPE ~ hDial

The simplest form of C# programming is the **Console Application**. Do FILE, NEW PROJECT, then select CONSOLE APPLICATION. You write your program and to test it:-

- BUILD is used to compile and like: BUILD SOLUTION
- DEBUG is used to execute it: DEBUG START DEBUGGING

```
// console application only ~ our copy we will call hDial.cs

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, ii;
            double lat, sinlat, lng, hla, hlat;
            float j, corh;
            string line;

            float fData = 12.345f; // no reason for this here at all, merely
            int iData = (int)fData; // a type conversion

            System.Console.WriteLine("*** Illustrating Shadows ~ hDial.cs in c# ***");
            System.Console.WriteLine("Enter a latitude: ");
            line = System.Console.ReadLine();
            lat = Convert.ToDouble(line);

            System.Console.WriteLine("Enter a longitude offset (+W -E): ");
            line = System.Console.ReadLine();
            lng = Convert.ToDouble(line);
            corh = (float)(lng * 4 / 60);

            for (i = -6; i < 7; i++) {

                if (i == 0) { System.Console.WriteLine(" "); };
                if (i == 1) { System.Console.WriteLine(" "); };

                sinlat = Math.Sin(lat * 2 * 3.1416 / 360);

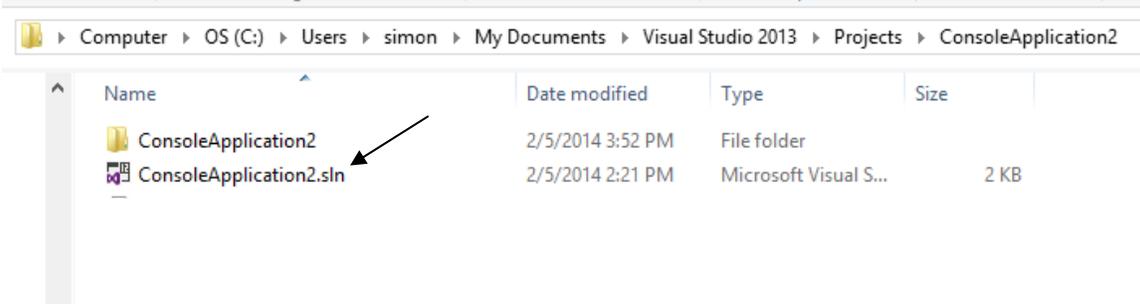
                // get the hour angle of the sun
                ii = (-1) * i;
                j = 15 * (ii + corh);

                /* get the resulting hour line angle ~ atan(sin(lat)*tan(hr*15) */
                hlat = sinlat * Math.Tan(j * 2 * 3.1416 / 360);

                /* get the hour line angle back to degrees */
                hla = 360 * (Math.Atan(hlat)) / (2*3.1416);

                System.Console.WriteLine("Hour: " + i + "hour line angle: " +
                    Math.Round(hla, 2, MidpointRounding.ToEven));
            }
            line = System.Console.ReadLine();
            System.Console.WriteLine("END");
        }
    }
}
```

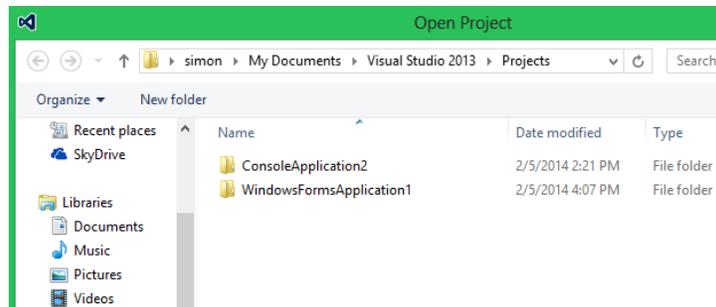
Once saved, you may locate your project is in:--



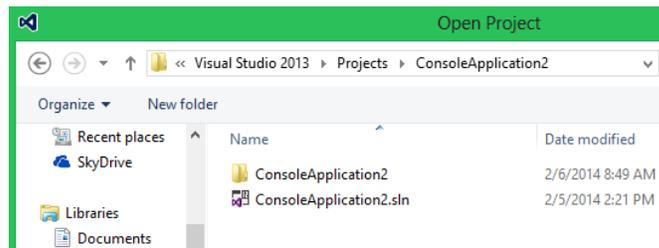
And your copy is only saved in your specified folders if you do a PUBLISH. PUBLISH is used for sending a final copy somewhere somehow. **NOTE:** You can save the above folders wherever you wish and restore them back in the event you screw things up badly. I do that.

If you wish to change your application's programming, do:-

FILE, OPEN PROJECT,

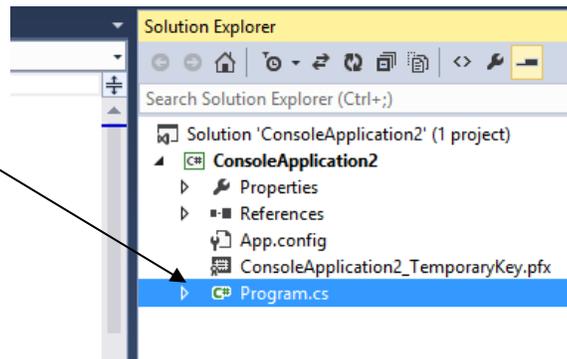


Then click on the desired project folder, and then select the .SLN file (SLN means Microsoft Visual Studio **S**olution)



NOTE: And if no source code comes up, then locate it as the .CS file

click the Program.cs and it will open up for you to mess with it. Program.cs is the main program. When using forms, then Program.cs invokes your program: Form1.cs etc



The above general principles work for the next two levels of C# programming. Those next two levels use forms. And sometimes when loading the project for changes, the graphical depiction of the form does not appear, only the coded version. The next section for Windows Form Application discusses those issues.

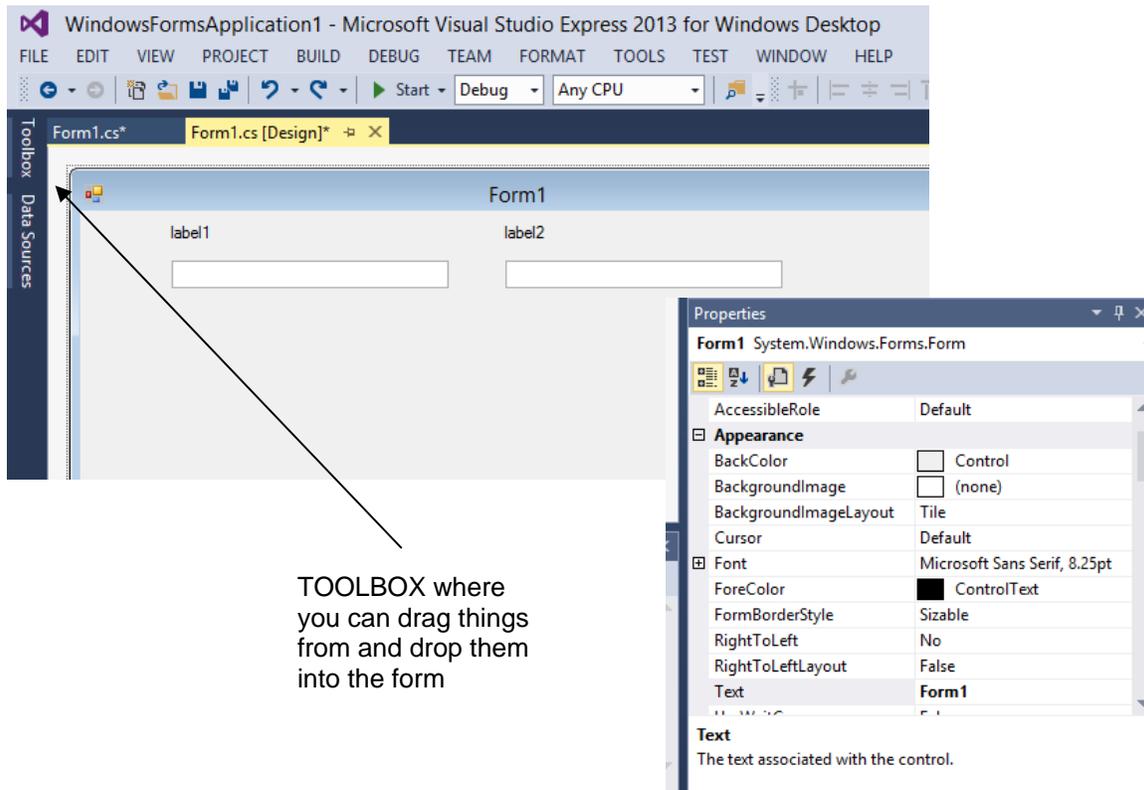
So, the next step is **Windows Form Application**, which is GUI based, but text only, which we will cover next, armed with the above helpful ideas.

[http://msdn.microsoft.com/en-us/library/360kwx3z\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/360kwx3z(v=vs.90).aspx)

And the next step after that will be WPF, **Windows Presentation Foundation (WPF)** application which allows graphics.

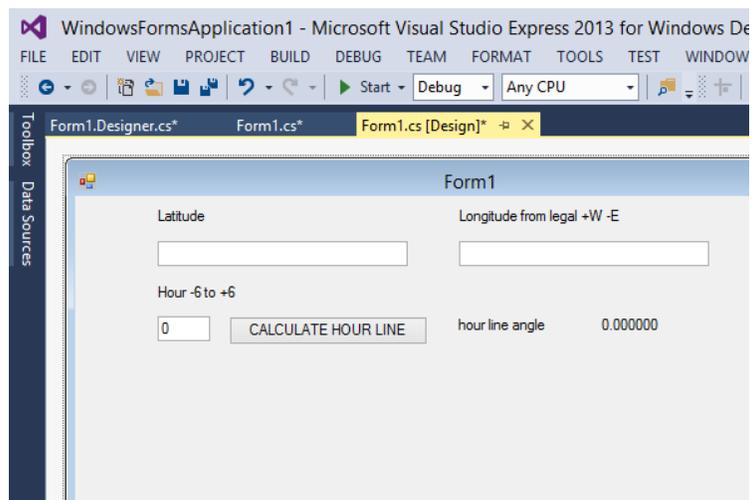
[http://msdn.microsoft.com/en-us/library/bb655895\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/bb655895(v=vs.90).aspx)

Windows Form Application, which is GUI based, but text only.



When you do a FILE and then NEW PROJECT, then Windows Form Application, a form will pop up. The process is similar to the Visual Basic process, and the Lazarus GUI process, so only key issues will be discussed here. As with VB and Lazarus, buttons trigger events and they in turn invoke user code. When a BUTTON is double clicked when in the form, a stub of code is generated to handle the event; this is where the guts of the program lie.

The final form for this application



The key labels are "lat", "lng", "hour" for input, with "hla" for output.

The following is the C# program using just forms.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    static class Program
    {
        /// The main entry point for the application.
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

↓
Form1.cs

```
using System; // This is Form1.cs, which is invoked by
using System.Collections.Generic; // Program.cs (see above inset)
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void calc_Click(object sender, EventArgs e)
        {
            // get lat and lng and calculate hla
            double dHLA = 100.000;
            string sHLA = "123.45";
            this.BackColor = Color.Yellow;

            double dLAT, dLNG, dHOUR;
            // string sLAT, sLNG, sHOUR;

            dLAT = Convert.ToDouble(lat.Text); // Get the data
            dLNG = Convert.ToDouble(lng.Text);
            dHOUR = Convert.ToDouble(hour.Text);
            dHOUR = dHOUR - (4 * dLNG / 60);

            dHLA = Math.Atan( Math.Sin(dLAT *2*3.1416/360) *
                            Math.Tan(15*dHOUR*2*3.1416/360) ) ;
            dHLA = 360 * dHLA / (2*3.1416);
            dHLA = Math.Round(dHLA, 3, MidpointRounding.ToEven);
            sHLA = Convert.ToString(dHLA);

            hla.Text = sHLA; // Put the data
        }
    }
}
```

This is the simplest form of C# programming for a TEXT FORM.

BUILD is used to compile and like: BUILD SOLUTION
 DEBUG is used to execute it: DEBUG START DEBUGGING

The screenshot shows a Windows form titled "Form1" with a white background and a green border. It contains the following elements:

- Two text boxes: "Latitude" and "Longitude from legal +W -E". Both are empty.
- A text box: "Hour -6 to +6" with the value "0".
- A button: "CALCULATE HOUR LINE".
- Text labels: "hour line angle" and "0.000000".

And the result it:-

The screenshot shows the same "Form1" window, but now with the following values:

- Latitude: 33.5
- Longitude from legal +W -E: 7.1
- Hour -6 to +6: 3
- The "CALCULATE HOUR LINE" button is highlighted with a blue border.
- hour line angle: 23.2518596495266

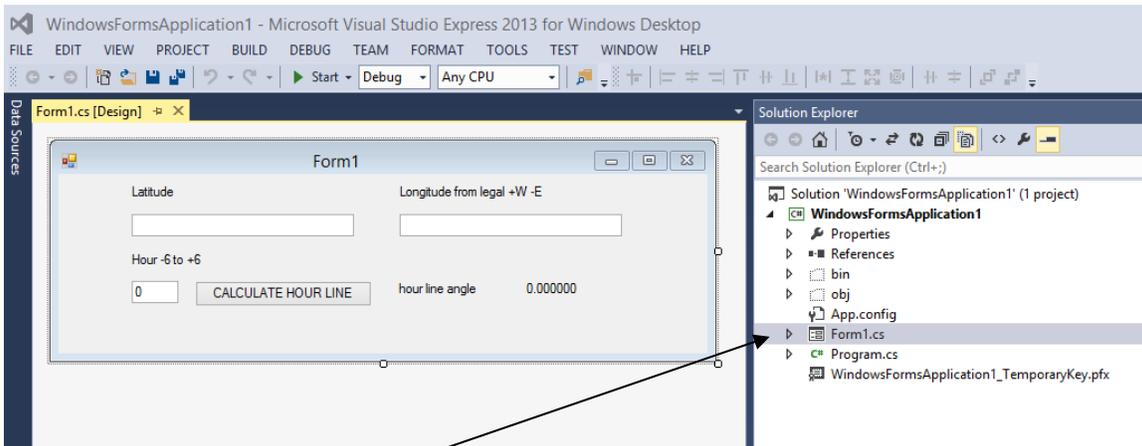
TIME OF DAY			Horizontal hour line angles with long corr	
am	pm		am	pm
12.00	12.00		3.93	-3.93
11.50	0.50		8.18	0.22
11.00	1.00		12.63	4.38
10.50	1.50		17.41	8.64
10.00	2.00		22.66	13.12
9.50	2.50		28.56	17.94
9.00	3.00		35.34	23.25
8.50	3.50		43.25	29.24
8.00	4.00		52.57	36.12
7.50	4.50		63.48	44.17
7.00	5.00		75.89	53.66
6.50	5.50		89.28	64.74
6.00	6.00		-77.28	77.28

These have longitude correction

For latitude 33.5 and longitude 112.1 from legal meridian of 105, we have the +7.1 longitude correction. Then, at 3pm we get an HLA of 23.25 degrees, which matches the Excel spreadsheet.

The general principles for console applications work for this (Windows Form Application) and the next level, Windows Presentation Foundation (WPF) of C# programming. These two levels use forms.

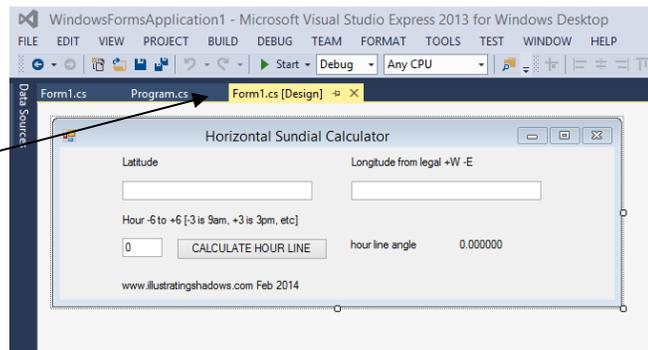
NOTE: Sometimes when loading the project for changes, the graphical depiction of the form does not appear, only the coded version. So, how does one get the graphical depiction to re-appear?



Locate the Form1.cs and click it even if the IDE shows Form1.cs as being there in the IDE. This will get Form1.cs [Design]

Whereas Program.cs was the main program in the console application, things are now different.

Form1.cs is actually the user program now, and with it should come up the form layout. With forms, the Program.cs is the code stub that invokes the Form code which is the main guts of the program.



NOTE: If that does not work, try VIEW on the top menu bar and then VIEW DESIGNER.

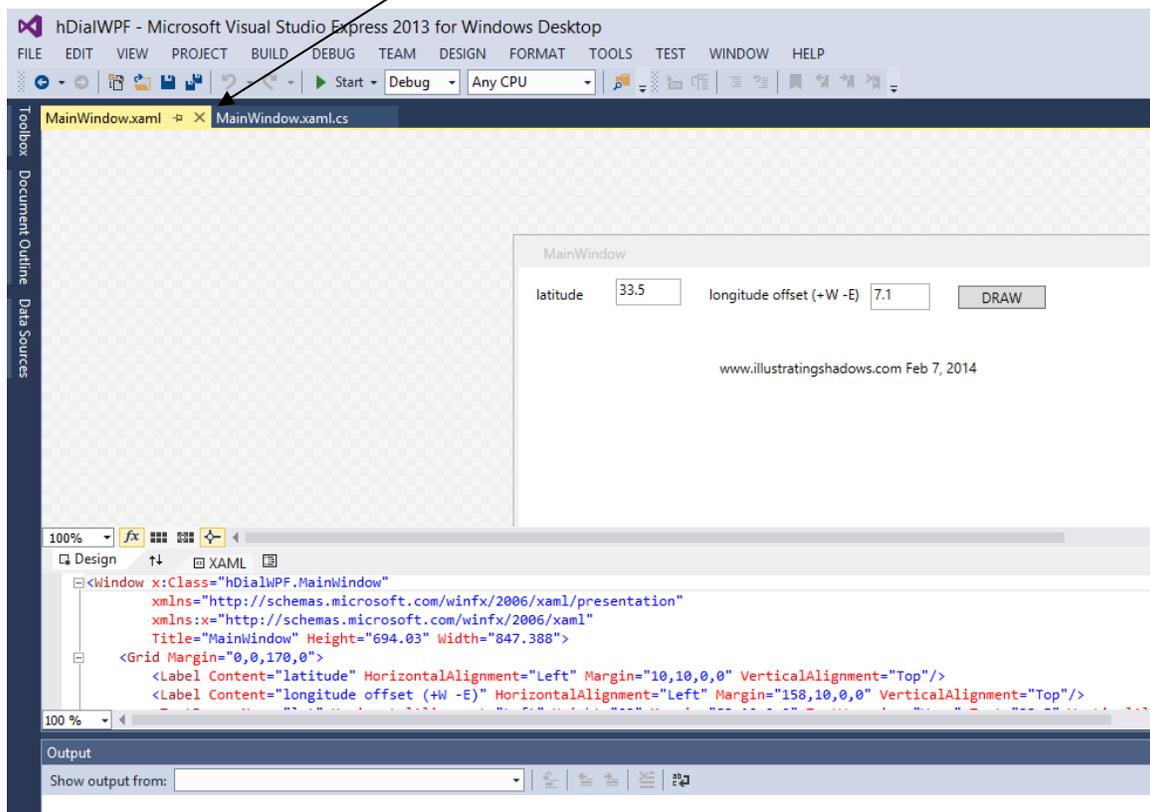
NOTE: Sometimes the forms design comes up but not its associated program. In that case use FILE, OPEN, and open the Form1.cs file.

Windows Presentation Foundation (WPF) application

The most advanced application for the PC is however the WPF. There are a couple of graphical systems provided with WPF, one is "stackpanel", the other is "canvas". Not being over supplied with documentation, I tried "stackpanel" first, it did what it implied, it stacked graphical objects one below the other. So then I tried "canvas" which I have used before with "Java Script". That worked well.

As before, a form was designed, and that will be also where the graphics is displayed, obliterating the buttons and so on.

The main program is called by default "MainWindow.xaml.cs"



The code was fairly straight forward, and since this is part of the PBE (programming by example) system, there are labels holding data, hour lines, and circles as well all displayed.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace hDialWPF
```

a FILE, NEW PROJECT, WPF Application generated by the IDE when you do

```

{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        private void drawDial_Click(object sender, RoutedEventArgs e)
        {
            // this is where we come when DRAW is clicked
            // in Windows Form Application we used:   this.BackColor = Color.Yellow;
            // In WPF (Windows Presentation Foundation) we use:
            this.Background = Brushes.Yellow;
            // http://stackoverflow.com/questions/979876/set-background-color-of-wpf-textbox-in-c-sharp-code

            // get lat and lng and calculate hla
            double dHLA, corrM;
            string lLat = "23.456";

            double dLAT, dLNG, dHOUR, sLat;
            // string sLAT, sLNG, sHOUR;

            dLAT = Convert.ToDouble(this.lat.Text); // Get the data
            dLNG = Convert.ToDouble(this.lng.Text);
            sLat = Math.Sin(dLAT * 2 * 3.1416 / 360);

            corrM = 4 * Convert.ToDouble(this.lng.Text);
            corrM = Math.Round(corrM, 2, MidpointRounding.ToEven);
            // the following works but is not displayed as the graph trashes the window
            this.corr.Content = Convert.ToString(corrM);
            this.corrNote.Content = "Minutes";

            double radius = 300.0;

            // do not use "stackpanel" as it does what it says, stacks graphical images
            // do use "canvas" http://www.c-sharpcorner.com/uploadfile/mahesh/canvas-in-wpf/
            Canvas myCan = new Canvas();

            for (int hr = -5; hr <= 5; hr++)
            {
                dHOUR = hr - (corrM / 60);
                dHLA = Math.Atan(sLat * Math.Tan(15 * dHOUR * 2 * 3.1416 / 360));
                dHLA = 360 * dHLA / (2 * 3.1416); // dHLA is now in degrees
                dHLA = Math.Round(dHLA, 3, MidpointRounding.ToEven);

                if (hr == 0)
                {
                    // at noon state the noon hour line angle
                    // the following works but is not displayed as graph trashes it
                    this.noonName.Content = "Noon hla:";
                    this.noonHla.Content = Convert.ToString(dHLA);
                }

                // http://msdn.microsoft.com/en-us/library/ms747393(v=vs.110).aspx
                // hla.Text = sHLA; // Put the data

                // Add a Line Element
                // NOTE: the web page's example differs on this line
                Line myLine = new Line();
                Label myHr = new Label();
                Label myHla = new Label();
                myLine.Stroke = System.Windows.Media.Brushes.Blue;
                myLine.X1 = 400; // dial center
                myLine.Y1 = 400;

                // NOTE: X2 is a displacement from X1, and ditto Y2
                myLine.X2 = myLine.X1 + (int)(radius * Math.Sin(dHLA * 2 * 3.1416 / 360));
                myLine.Y2 = myLine.Y1 - (int)(radius * Math.Cos(dHLA * 2 * 3.1416 / 360));
                myLine.StrokeThickness = 1;
            }
        }
    }
}

```

also IDE generated

```

myHr.Content = Convert.ToString(hr+12);
Canvas.SetLeft(myHr, myLine.X2-10);
Canvas.SetTop (myHr, myLine.Y2-25);

myHla.Content = Convert.ToString(Math.Round(dHLA,
    2, MidpointRounding.ToEven));
Canvas.SetLeft(myHla, myLine.X2 - 10);
Canvas.SetTop (myHla, myLine.Y2 - 45);

Ellipse dot = new Ellipse();
dot.Height = 5; dot.Width = 5; dot.StrokeThickness = 1;
dot.Stroke = System.Windows.Media.Brushes.Blue;
Canvas.SetLeft(dot, myLine.X2);
Canvas.SetTop (dot, myLine.Y2);

myCan.Children.Add(myLine);
myCan.Children.Add(myHr);
myCan.Children.Add(myHla);
myCan.Children.Add(dot);
}

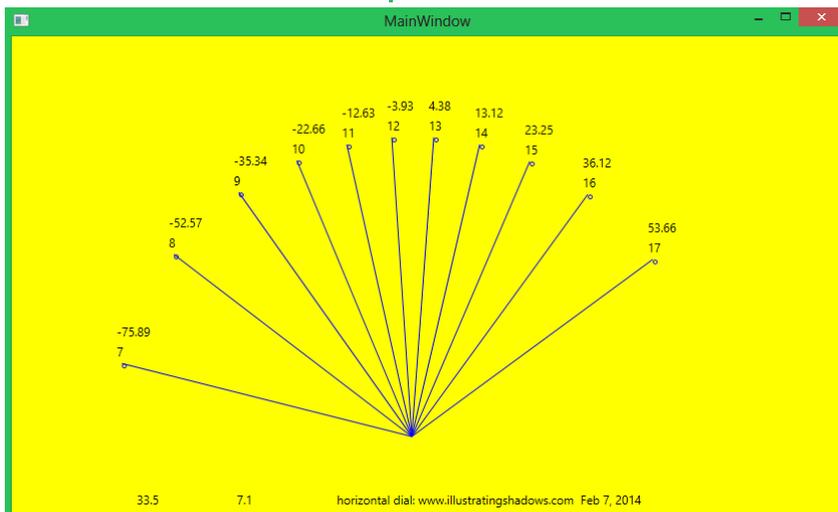
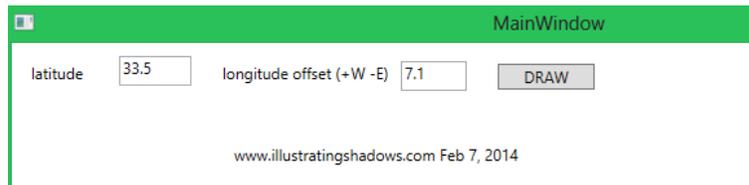
Label details = new Label();
Canvas.SetLeft(details, 120);
Canvas.SetTop (details, 450);
details.Content = this.lat.Text;
myCan.Children.Add(details);

Label offset = new Label();
Canvas.SetLeft(offset, 220);
Canvas.SetTop(offset, 450);
offset.Content = this.lng.Text;
myCan.Children.Add(offset);

Label who = new Label();
Canvas.SetLeft(who, 320);
Canvas.SetTop(who, 450);
who.Content = "horizontal dial: www.illustratingshadows.com Feb 7, 2014";
myCan.Children.Add(who);

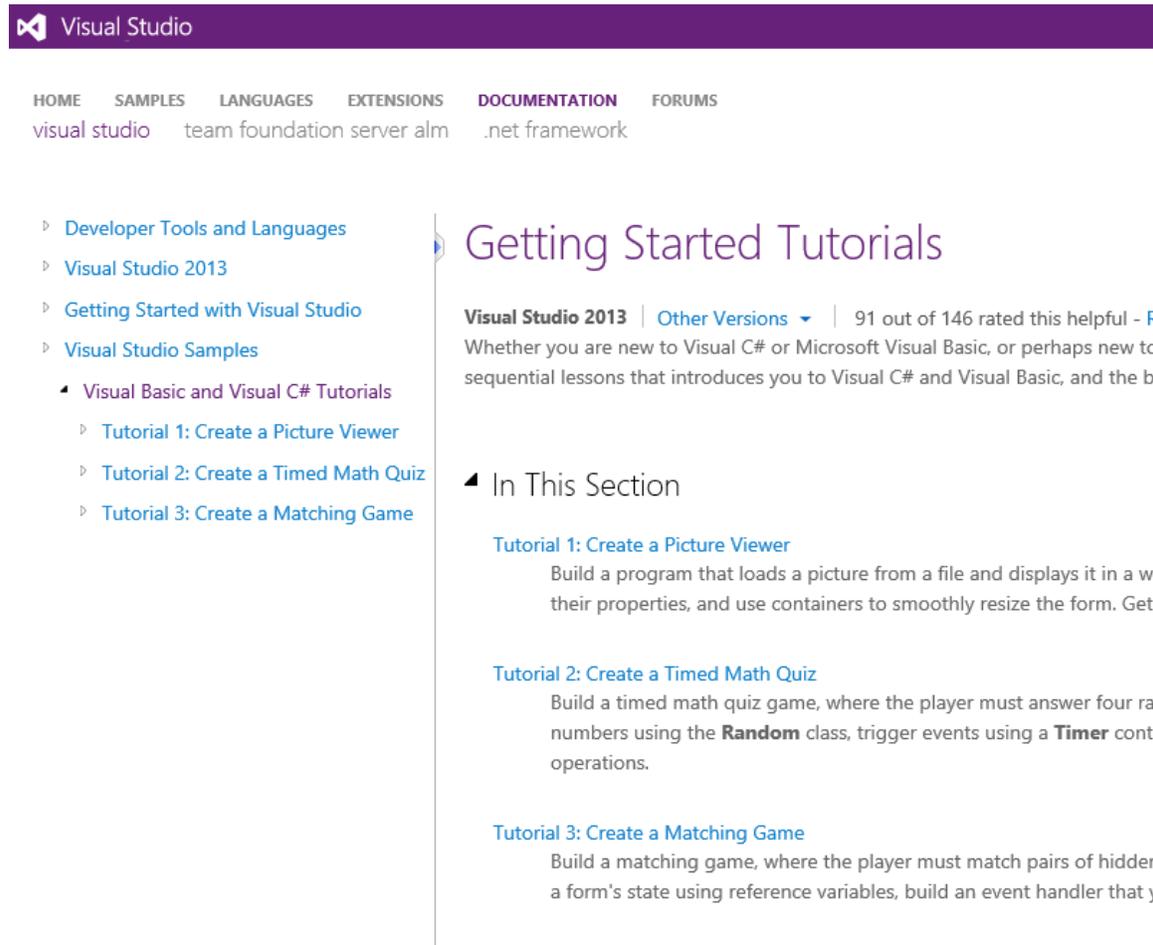
this.Content = myCan;
}
}
}

```



And finally, the online tutorial:-

[http://msdn.microsoft.com/library/vstudio/dd492171\(v=vs.120\)](http://msdn.microsoft.com/library/vstudio/dd492171(v=vs.120))



The screenshot shows the Visual Studio documentation website. At the top, there is a purple navigation bar with the Visual Studio logo and the text 'Visual Studio'. Below this, a horizontal menu contains links for 'HOME', 'SAMPLES', 'LANGUAGES', 'EXTENSIONS', 'DOCUMENTATION', and 'FORUMS'. Under 'DOCUMENTATION', there are sub-links for 'visual studio', 'team foundation server', 'alm', and '.net framework'. On the left side, a vertical navigation pane lists several categories: 'Developer Tools and Languages', 'Visual Studio 2013', 'Getting Started with Visual Studio', 'Visual Studio Samples', and 'Visual Basic and Visual C# Tutorials'. The 'Visual Basic and Visual C# Tutorials' category is expanded to show three sub-tutorials: 'Tutorial 1: Create a Picture Viewer', 'Tutorial 2: Create a Timed Math Quiz', and 'Tutorial 3: Create a Matching Game'. The main content area on the right is titled 'Getting Started Tutorials' and includes a rating of '91 out of 146 rated this helpful - R'. Below the title, there is a section 'In This Section' which lists the three tutorials with brief descriptions: 'Tutorial 1: Create a Picture Viewer' (Build a program that loads a picture from a file and displays it in a window, and use containers to smoothly resize the form. Get their properties, and use containers to smoothly resize the form. Get), 'Tutorial 2: Create a Timed Math Quiz' (Build a timed math quiz game, where the player must answer four random numbers using the **Random** class, trigger events using a **Timer** control), and 'Tutorial 3: Create a Matching Game' (Build a matching game, where the player must match pairs of hidden cards, build a form's state using reference variables, build an event handler that y).