Windows Forms GUIs 1: Example code and tutorial

Building Graphical User Interfaces (GUIs)

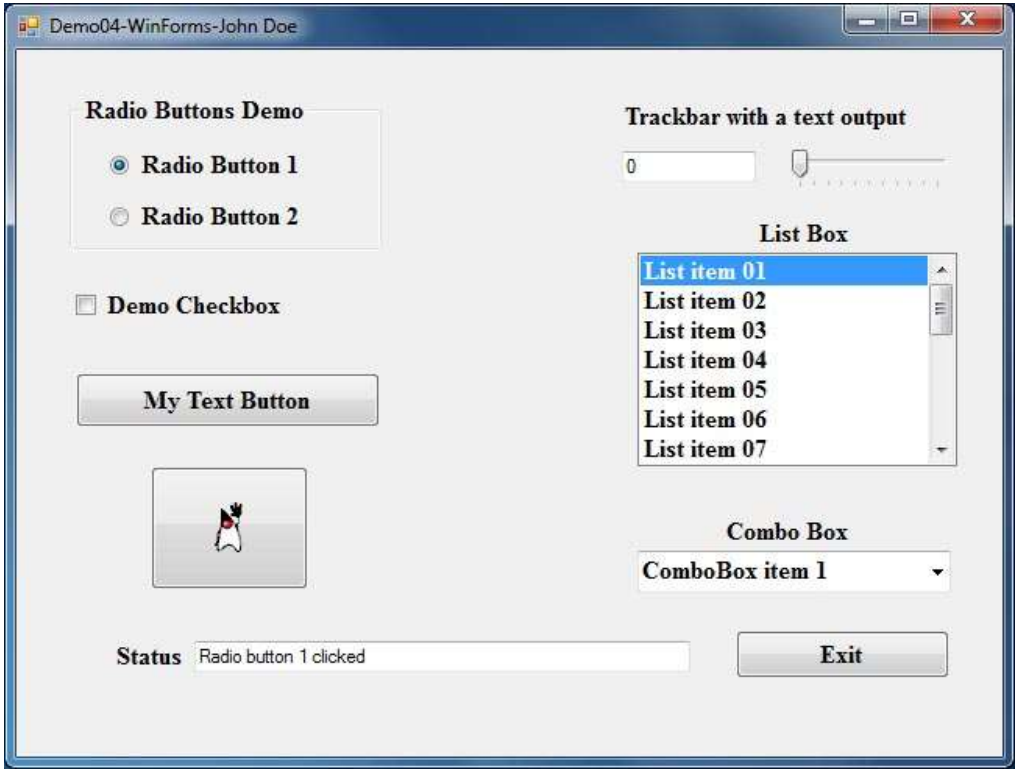
Using Windows Forms

All of these examples assume that Microsoft Visual C++ 2017 is the compiler being used. Based on a tutorial written by Dr. Rick Coleman.

There is a bug in Visual Studio 2017 that will result in you seeing an error when you first make the project.

Exercise 4: A Windows Forms application

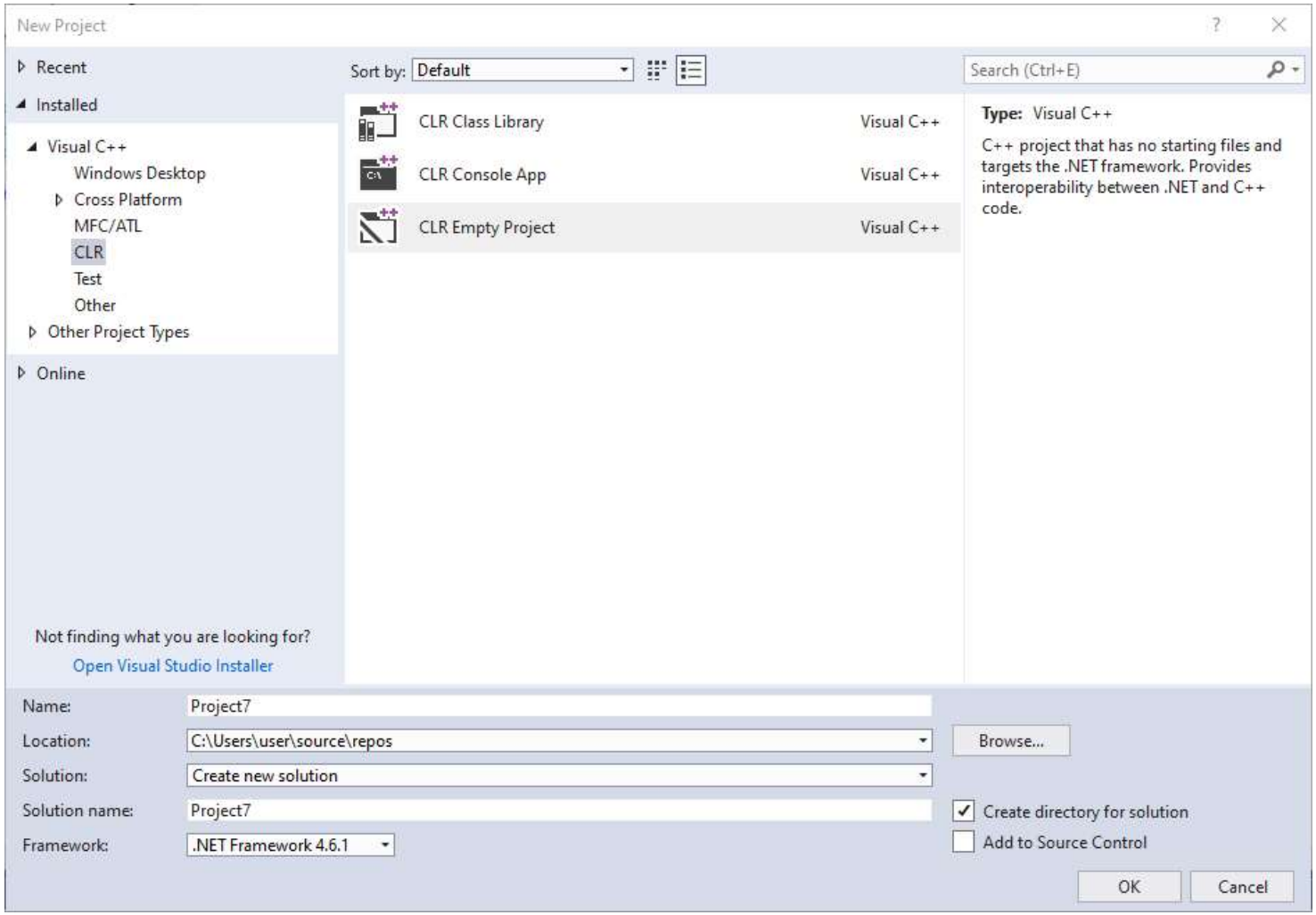
In this application you are going to build a Windows Forms application with a look similar to the MFC Dialog based application.



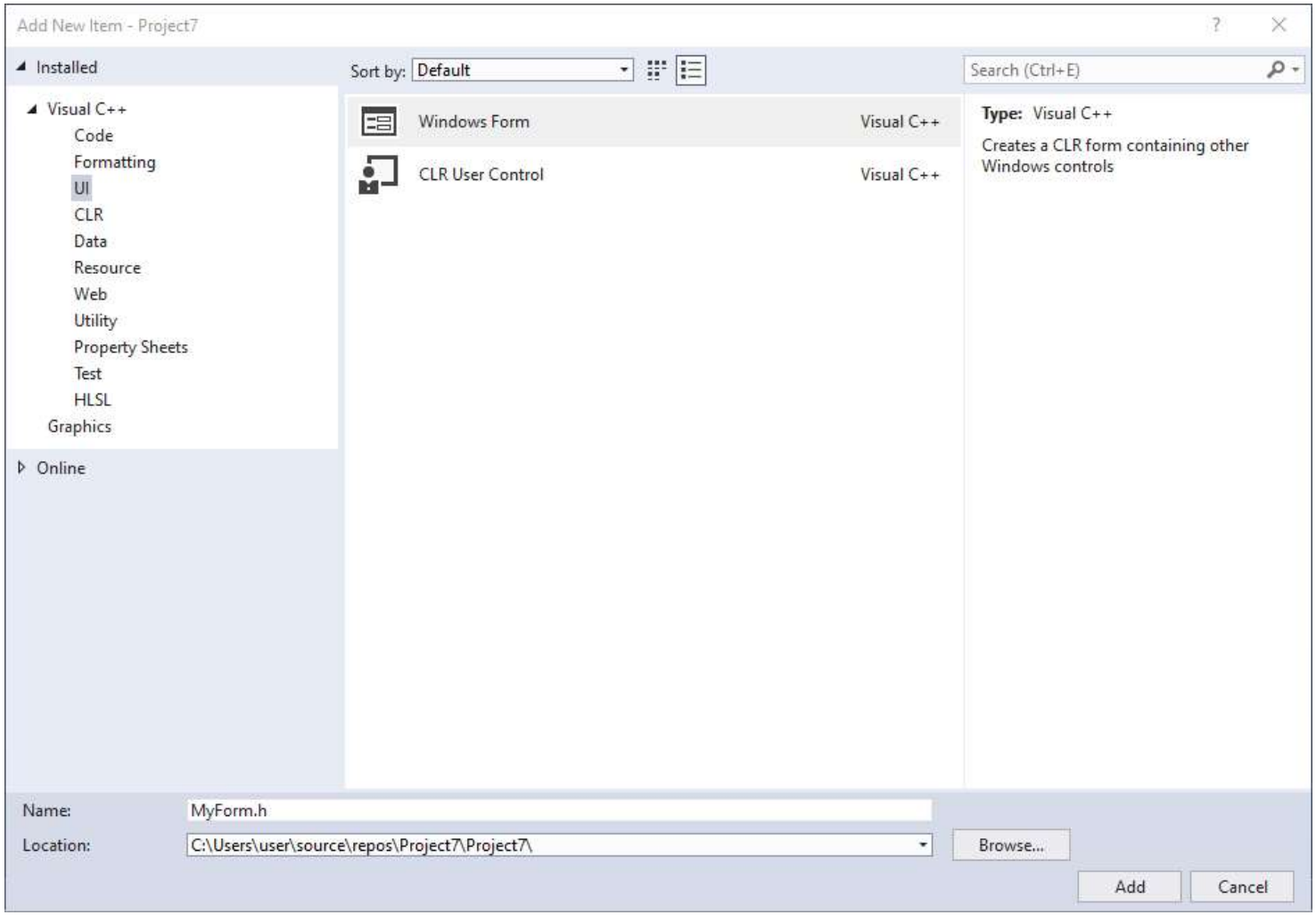
Creating the project

Here we will go through the steps to create the project and add the Windows Form to the project.

1. Create a new Visual Studio project of type "CLR Empty Project." It should be available under the "CLR" category as shown here.



2. In the newly created project, right click on "Header Files" in the Solution Explorer and select the Add->New Item option. In the Add New Item dialog, add a new "Windows Form" from the "UI" category as shown here.



3. This will probably result in Visual Studio trying to bring up the designer and showing an error. We will now fix this error. To fix it, open the "MyForm.cpp" file in the code editor by double clicking it in the Solution Explorer. Probably all that's in that file is something like this:

```
#include "MyForm.h"
```

We'll need to add some additional stuff to this file so it looks like this (note: for the namespace in the "using namespace" line, you'll need to replace that with the name of your project -- mine was Project8):

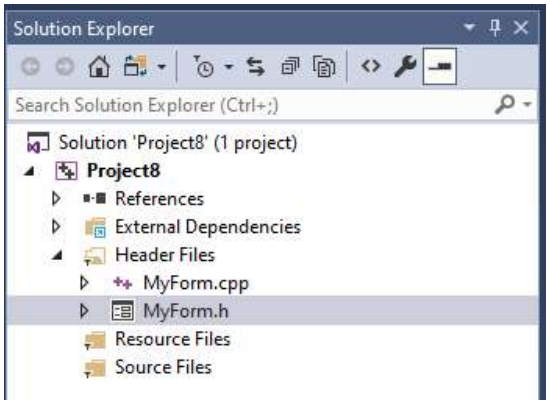
```
#include "MyForm.h"
#include <Windows.h>

using namespace Project8;

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
```

```
{

        Application::EnableVisualStyles();
        Application::SetCompatibleTextRenderingDefault(false);
        Application::Run(gcnew MyForm());

        return 0;

}
```

4. After adding this code, close and reopen the design view for the form. Close the old one by clicking the "x" to close it. Open a new one by double clicking on the "MyForm.h" file in the Solution Explorer.

The Files Visual Studio Created

Take a look at the files Visual Studio created automatically for you when you created the project and added the Windows Form.



MyForm.h

Right click this file in the Solution Explorer and select **View Code**

- This file is where Visual Studio automatically writes code as you build your application. Note that all this code is placed in a **.h** file which goes completely against the conventional C++ programming style of putting code in a **.cpp** file. In **MyForm.h** there are two functions you should notice:
    1. **The Constructor**. Note that this first calls **InitializeComponent()** and then has a comment "//TODO: Add the constructor code here". This is where you want to add your code to initialize any components and/or member variables. For example, here is where you would set the initial selected index of items in list boxes or combo boxes. Do not attempt to do it in the InitializeComponent() function (which would seem to be the logical place to do it) because every time you make a change in the form using the form designer it rewrites this function and will overwrite any initializations you put there.

2. **InitializeComponent()**. As you add and modify widgets in the Form Designer window Visual Studio modifies this function to add the code to create and set the properties of all the widgets.
3. **Event Handlers**. As you add event handler functions for each of the widgets in your Forms GUI they will be created for you below the **InitializeComponent()** function.

Building the GUI

If the Form Editor / Designer is not already open showing MyForm, open it by double clicking MyForm.h in the Solution Explorer window.

Look at the left or right edge of the Visual Studio window and find where the **Toolbox** tab is located. When you click this it expands to display all the widgets you can add to the form. There are a number of catagories of widgets. You want to expand the **All Windows Forms** widgets.

You can now build your GUI by clicking a widget in the list then clicking in the form where you would like to creat the widget. Below you will find notes on how to create the demonstration form shown above. You may copy it or feel free to experiment with your own layout and design.

Note: You will add the radio buttons and group widgets last as this can sometimes cause the errors mentioned above.

Build the GUI

MyForm window

1. In the Properties list change the **Text** Property (window title) to the name of your project and replace **John Doe** as seen in the title of the above example with your name.
2. Set the **Size** Property to 640 x 480

CheckBox

1. Create a CheckBox by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Text** Property to "Demo Check Box"
3. Set the **Font** Property to Times New Roman, bold, 12
4. Double click the CheckBox to create an event handler function in MyForm.h.

Text Button

1. Create a Button by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Text** Property to "My Text Button"
3. Set the **Font** Property to Times New Roman, bold, 12
4. Double click the Button to create an event handler function in MyForm.h.

Image Button

1. You will need to prepare an image for the button. You can create an icon resource just as you did in the previous exercise, or copy any **.ico** file from another directory into the project directory. You can also use an image (.JPG, .GIF, .BMP, .PNG). Just copy it into your project directory. If you use an image file it must already be the correct size to fit on the button. It will not be resized by Visual Studio.
2. Create a button by selecting it in the Toolbox then click and drag on the form to create and size the button.
3. Select the **Image** property then click in the right column to show the select image dialog box. An Open File dialog box will appear that will let you select the image or icon file to use. (You may see a different type of dialog box in which you will have to (1) click the "Local Resource" radio button, (2) click the "Import" button, (3) select "All Files" in the combobox in the lower right of the Open File dialog box that appears, (4) Use this Open File dialog box to select the icon or image to load.)
4. Double click the Button to create an event handler function in MyForm.h.

"Status" Label

1. Create a Label by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Text** Property to "Status"
3. Set the **Font** Property to Times New Roman, bold, 12

TextBox for status output

1. Create a TextBox next to the Status label by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Font** Property to Times New Roman, bold, 12

"TrackBar with a text output" Label

1. Create a Label by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Text** Property to "TrackBar with a text output"
3. Set the **Font** Property to Times New Roman, bold, 12

TextBox for trackbar output

1. Create a TextBox by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Font** Property to Times New Roman, bold, 12

TrackBar

1. Create a TrackBar by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Maximum** Property to 100
3. Set the **TickFrequency** Property to 10
4. Double click the TrackBar to create an event handler function in MyForm.h.

ListBox

1. Create a ListBox by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Font** Property to Times New Roman, bold, 12
3. Double click the ListBox to create an event handler function in MyForm.h.

ComboBox

1. Create a ComboBox by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Font** Property to Times New Roman, bold, 12
3. Double click the ComboBox to create an event handler function in MyForm.h.

Exit Button

1. Create a Button by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Text** Property to "Exit"
3. Set the **Font** Property to Times New Roman, bold, 12
4. Double click the Button to create an event handler function in MyForm.h.

Do this last because of the circular error problem that is a Visual Studio bug.

GroupBox

1. Create a GroupBox by selecting it in the Toolbox then click on the form where you want to create it.
2. Set the **Text** Property to "Radio Button Demo"
3. Set the **Font** Property to Times New Roman, bold, 12

Two RadioButtons

1. Create two RadioButtons in the GroupBox by selecting RadioButton the Toolbox then click inside the boundary of the GroupBox to create each one.
2. Set the **Text** Properties to "Radio Button 1" and "Radio Button 2"
3. Set the **Font** Property of each to Times New Roman, bold, 12
4. Double click each RadioButton to create an event handler function in MyForm.h.

Add code to add items to the list box and the combobox

1. **List box -** Look in the constructor for MyForm and add code after the call to InitializeComponent() to add items to the list box.

```
public ref class MyForm : public System::Windows::Forms::Form
{
public:
        MyForm(void)
        {
                InitializeComponent();
                //
                //TODO: Add the constructor code here
                //
                this->listBox1->Items->Add("List item 01");
                this->listBox1->Items->Add("List item 02");
                this->listBox1->Items->Add("List item 03");
                this->listBox1->Items->Add("List item 04");
                this->listBox1->Items->Add("List item 05");
                this->listBox1->Items->Add("List item 06");
                this->listBox1->Items->Add("List item 07");
                this->listBox1->Items->Add("List item 08");
                this->listBox1->Items->Add("List item 09");
```

```
this->listBox1->Items->Add("List item 10");
this->listBox1->SelectedIndex = 0;
```

2. **Combo box -** Look in the constructor for MyForm and add code after the call to InitializeComponent() to add items to the combo box.

```
this->comboBox1->Items->Add("Combobox item 01");
this->comboBox1->Items->Add("Combobox item 02");
this->comboBox1->Items->Add("Combobox item 03");
this->comboBox1->Items->Add("Combobox item 04");
this->comboBox1->Items->Add("Combobox item 05");
this->comboBox1->SelectedIndex = 0;
```

Add code to the event handlers

1. **RadioButtons, CheckBox, text Button, and icon Button**
   1. Look in MyForm.h close to the top and you should see a list of variable names for each of the widgets created. Look for the one for the "Status" TextBox and note the name of the widget (**textBox1** in this example):

```
private: System::Windows::Forms::TextBox^  textBox1;
```

   2. Find the event handler function for each of the named widgets above and add a line similar to the one below, which is for the first radio button.

```
this->textBox1->Text = "Radio button 1 clicked";
```

2. You can also do a check to determine if the check box is checked with:

```
if(this->checkBox1->Checked)
```

- **List box**
  Find the event handler for the list box and add the following code to display the selected line in the status text box

```
this->textBox1->Text =  (System::String ^)(this->listBox1->Items[this->listBox1->SelectedIndex]) + " selected";
```

- **Combo box**
  Find the event handler for the combo box and add the following code to display the selected line in the status text box

```
this->textBox1->Text = (System::String ^)(this->comboBox1->Items[this->comboBox1->SelectedIndex]) + " selected";
```

- **TrackBar**
  1. Look in MyForm.h close to the top and you should see a list of variable names for each of the widgets created. Look for the one for the trackbar text box and note the name of the widget (textBox2 in this example).

```
private: System::Windows::Forms::TextBox^  textBox2;
```

  2. Find the event handler function for the TrackBar and add the line below

```
this->textBox2->Text = this->trackBar1->Value.ToString();
```

- **Exit button**
  Find the event handler for the Exit button and add the following code to terminate the application when it is clicked.

```
this->Close();
```

Compile and run your application. Play with the widgets to see how they work. Try adding and experimenting with other widgets

For more information on any of the widgets try doing a Google search on the widget name **+"windows forms"**.