**Developing wxWidgets Applications**

with

Visual Studio 2015/2017[[1]](#footnote-1), [[2]](#footnote-2), [[3]](#footnote-3)

and wxFormBuilder

**Overview**: Before getting started, you may want read some background material about developing application with a graphical user interface (GUI). You can find such an overview at <http://people.cedarville.edu/Employee/kshomper/cs1220_web/lecture_notes/gui_intro.htm>.

**Using wxWidgets**: wxWidgets is one of several GUI development libraries which allow programmers to add a modern GUI interface to their C++ applications. However, before you can use wxWidgets, it needs to be installed on your computer or laptop[[4]](#footnote-4). If wxWidgets is not installed, follow the directions in the following paragraph; otherwise, skip ahead to the section entitled *Using Visual Studio*.

* **Installation**: To install wxWidgets on your laptop or desktop computer:

1. Use [filr](https://filr.cedarville.edu/) to copy the file *wxPack\_v2.8.12.01.exe* from

*S:\DEPT\EG\Computer Science\wxWidgets* to your computer.

1. Double-click on *wxPack\_v2.8.12.01.exe* to install wxWidgets.
   1. Click *Run*, if prompted.
   2. Click *Yes*, if prompted.
   3. Click *Next*.
   4. Click *I accept the agreement* and click *Next*.
   5. In the drop down box (where it says *Full Installation*) choose *Visual C++ Only Runtime Version 10.0*[[5]](#footnote-5) and click *Next*.
   6. Click *Next* to create a start menu folder.
   7. By default, wxWidgets does **not** install in C:\Program Files. If you want wxWidgets to install here, click the *Browse* button and select this folder[[6]](#footnote-6), then click *Next*.
   8. Click *Install*.
   9. Click *Finish*.

**Using Visual Studio**: Although wxWidgets is a multi-platform widget library, for this project we will use it within the Windows family of operating systems (Windows7, Windows10, etc.). Therefore, we will also be using Visual Studio as our development environment. Visual Studio provides good integration with wxWidgets, if you’ve installed wxWidgets as specified above[[7]](#footnote-7).

* **The First wxWidgets Project**: To get started with your wxWidgets application, copy the FormBuilderApps folder from S:\DEPT\EG\Computer Science\CS1220 to a folder of your own choosing on your laptop (or other computer).
  1. Now, with the above destination folder opened in Windows Explorer, double click on the file FormBuilderApps.sln.
  2. If you are using Visual Studio 2015 or 2017 and are prompted to update the project, go ahead and click OK. If you are using VS2017 see the next to last page in this document for additional instructions.
  3. From within Visual Studio, in the Solution Explorer, you should now see a new Visual Studio C++ project called wxWidgetsApp with five folders:
     + External Dependencies,
     + Header Files,
     + Resource Files,
     + Source Files, and
     + wxFormBuilder Files.
* **Project Folder Structure**: Although, there are five folders associated with the Visual Studio wxWidgets project, you only really care about the first three described below (the other two are described for completeness, but we do not use them). The five folders are:
  1. *Header Files*: In this folder we keep the project header files are not fully generated by wxFormBuilder (see below). These files are the specification for the application class, [*wxWidgetsApp.h*](http://people.cedarville.edu/Employee/kshomper/cs1220_web/examples/wxWidgetsApp.h) and *SubGUIClass.h* (not yet created)
  2. *Source Files:* In this folder we keep the project source files which are fully generated by wxFormBuilder (see below). These files are the implementation for the application class, [*wxWidgetsApp.cpp*](http://people.cedarville.edu/Employee/kshomper/cs1220_web/examples/wxWidgetsApp.cpp) and *SubGUIClass.cpp* (not yet created).
  3. *wxFormBuilder Files*: The wxFormBuilder files are where we keep the wxFormBuilder project file (*.fbp* extension, not yet created) which defines the GUI and the fully generated *GUIClass.h/.cpp* files.
  4. *Resource Files*: Our project does not use resources. This folder remains empty.
  5. *External Dependencies*: Our project will set all necessary dependencies automatically on compilation; therefore, you can safely ignore this folder.
* **The Application Class Files**: The files [*wxWidgetsApp.cpp*](http://people.cedarville.edu/Employee/kshomper/cs1220_web/examples/wxWidgetsApp.cpp) and [*wxWidgetsApp.h*](http://people.cedarville.edu/Employee/kshomper/cs1220_web/examples/wxWidgetsApp.h) which appear in the *Source* and *Header Files* folders, respectively, are the template application files. We’ll have more to say about these later.
* **Building the GUI**: In this step you will use wxFormBuilder to create the GUI for your application. Detailed instructions for building the GUI are beyond the scope of this document. However, below you’ll find a brief *Getting Started* which lists the things to keep in mind as you build the GUI. Also, during one of your CS1220 class periods, your instructor will demonstrate building a GUI using wxFormBuilder.
  1. **Getting Started with wxFormBuilder:** Begin by starting wxFormBuilder (e.g., Start … All Programs … wxPack … wxFormBuilder … wxFormBuilder).
     + First, because we’ll be using wxFormBuilder to generate the GUI class code, we want to set a few project properties immediately, so the code is generated in our *wxWidgetsApp* project folder. Therefore, to start, set the following properties (the property sheet appears on the right side of the application window):
       1. **name:** Set this value to “Sub.”
       2. **file:** This property’s value determines the filename of the generated code. Since we are generating code for the GUI Class, an appropriate value here would be “GUIClass.”
     + Next, save the wFormBuilder project file by choosing *File … Save As*.

1. In the file dialog that appears, fill in the *File name* to whatever you wish (I used *ttt.fbp*).
2. Ensure you save the file in the *wxWidgetsApp* folder in the Visual Studio solution you copied in the step *The First wxWidgets Project* above[[8]](#footnote-8).
   * + Finally, we will create the main frame (top-level window) for the GUI class to test the GUI class code generation. Do the following:
       1. Select the *Forms* tab.
       2. Click on the first icon (it looks like a tiny application window) appearing below the tabs. This will cause a top-level window to appear in the center of the screen.
       3. Change the *name* property to GUIClass.
       4. Click the gear-shaped icon in the tool bar, or (or select *File … Generate Code*, or press the *F8* key) to generate the GUI class.
       5. Select *Tools … Generate Inherited Class* (or press the *F6* key), select the checkbox, and press *OK* to generate a subclass for the GUI class.

If after following the above steps, you have *.h* and *.cpp* files for the application class (called *wxWidgetsApp.h/cpp*), the GUI class (*GUIClass.h/cpp*), and a subclass of the GUI class (*SubGUIClass.h/cpp*) in your Visual Studio *wxWidgetsApp* project folder, then your wxFormBuilder project is set up correctly.

* **Adding the wxFormBuilder Files to the Visual Studio Project:** After generating the GUI class files we need add this code to the Visual Studio project. Add the code as follows:
  + From the Visual Studio *Solution Explorer*, right-click on the *Header Files* folder, choose Add … Existing Item, and select the file *SubGUIClass.h*.
  + From the Visual Studio *Solution Explorer*, right-click on the *Source Files* folder, choose Add … Existing Item, and select the file *SubGUIClass.cpp*.
  + From the Visual Studio *Solution Explorer*, right-click on the *wxFormBuilder Files* folder, choose Add … Existing Item, and select the *GUIClass.h/.cpp* files. Also select the wxFormBuilder file (its extension will be *.fbp*).

* **Important Notes Regarding the GUI Subclass Files:** The code for the GUIclass (i.e., *GUIClass.h/cpp*) determines how the GUI is drawn. Since we regenerate the GUIclass each time we make changes to the GUI with wxFormBuilder, we DO NOT want to make custom changes to the GUIclass event handlers, because those changes will be overwritten when the GUIclass is regenerated. Rather, wxFormBuilder encourages developers to subclass the GUIclass and override the event handlers in the subclass. Above, we discussed how to automatically create a GUI subclass (e.g., select *Tools … Generate Inherited Class* or press the *F6* key which created the files *SubGUIClass.h/cpp*). Once you begin to edit these sub-classed files, you should *not* automatically regenerate them, or you may overwrite your custom changes.

1. **Making the Application Class Reference the GUI Subclass:** The last step in building the project is to make the application class, *wxWidgetsApp*, reference the GUI subclass. We do this by modifying the source file *wxWidgetsApp.cpp*. Open *wxWidgetsApp.cpp* in Visual Studio and edit the defined values for FILENAME and CLASSNAME (they appear lines 10 and 11, respectively). These two defines should be set as follows:
   * **FILENAME:** replace xxxxx with SubGUIClass.h (keep the quote marks)
   * **CLASSNAME:** replace XXXXX with SubGUIClass
2. **Testing the Project:** To test the project in Visual Studio select *Debug … Start Without Debugging* or press *CTRL-F5*.
   * AT THIS POINT THE PROJECT SHOULD COMPILE, BUT REPORTS MULTIPLE LNK2038 ERRORS. See the next page of instructions, Building a Visual Studio 2015/17 Project w/wxWidgets, for resolving these errors.

**Additional Help**: Assuming that wxWidgets was installed as specified above, the following applications and help are available from the Windows Start menu.

* **Start … All Programs … wxPack … wxWidgets Compiled … wxWidgets Help:** Comprehensive help documentation for using the wxWidgets library.

1. **Start … All Programs … wxPack … wxFormBuilder:** Access to the wxFormBuilder application.

**Building a Visual Studio 2015/17 Project w/wxWidgets**

**Issue**: The version of wxWidgets supplied with wxPack on the S: drive has been compiled with Visual Studio 2010 and not for later editions of Visual Studio (e.g., such as VS2015 or CS2017). Therefore VS2015/17 users will see LINK errors of the following type:

1>GUIClass.obj : warning LNK4075: ignoring '/EDITANDCONTINUE' due to '/SAFESEH' specification

1>wxbase28d.lib(baselib\_appbase.obj) : error LNK2038: mismatch detected for '\_MSC\_VER': value '1600' doesn't match value '1900' in GUIClass.obj[[9]](#footnote-9)

The warning you can safely ignore, but the errors will need to be corrected. To correct the errors follow the directions the “Solutions” paragraph below.

**Solutions**: The solution to this problem is to copy the libraries you need for the VS2015 compiler, a full description appears below.

Solution Process: Copies of the libraries you need are in the folder *S:\DEPT\EG\Computer Science\wxWidgets*. At this location is a subfolder *wxWidgets2.8.12 VS2015 Libraries*, double click on that subfolder. All of the files in the subfolder need to be copied and pasted to the folder *C:\Program Files (x86)\wxWidgets2.8\lib\vc\_lib*. These files will overwrite those that are already there. Once you have done this, rebuild your application.

**Developing wxWidgets Applications under Ubuntu Linux**

**Using wxWidgets**: wxWidgets and wxFormBuilder are already installed on john, so no installation is required. However, because they display a GUI, you MUST have XMing running locally on your PC or laptop and have PuTTY properly configured for X11. Instructions for doing this can be found on the [here](http://people.cedarville.edu/Employee/kshomper/cs1210_web/lecture_notes/login_instructions.htm) in paragraphs “Using PuTTY” and “Using XMing.”

**Developing Your C++ Code:** You have complete flexibility over the structure of your project in the linux environment whether you use *vi*, *gedit* or, *xCode* to create your program source on john or transfer your code from your own laptop using and ftp program like *FileZilla*. However you create your code, I recommend you put it in its own project directory.

Begin by copying the [*wxWidgetsApp.h*](http://people.cedarville.edu/Employee/kshomper/cs1220_web/examples/wxWidgetsApp.h) and [*wxWidgetsApp.cpp*](http://people.cedarville.edu/Employee/kshomper/cs1220_web/examples/wxWidgetsApp.cpp) files to your project directory.

Follow the notes on **Building the GUI** on page three of this document. To start wxFormBuilder on john, type: “wxformbuilder &” Also, read **Important Notes Regarding the GUI Subclass Files** and **Making the Application Class Reference the GUI Subclass** on page four.

**Windows versus Linux Issues:** If you developed your code in Visual Studio, it will need modifications to run on john. Common changes are listed in the paragraphs below. However, BEFORE making changes to your code, please READ the NOTE just below paragraph 4.

1. The *wxMessageDialog* class and *wxMessageBox()* macro are not automatically included on john, so if you used either of these, *wx/msgdlg.h* must be explicitly included.
2. If your SubGUIClass implementation uses *std::string* to set text in dialogs or labels on buttons and you get compilation errors, then replace *std::string* with *wxString*.
3. john’s wxWidgets libraries support Unicode characters, so string literals, e.g., *"like this"* will NOT compile automatically, so replace string literals with one of the following:
   1. The macro: *\_T("like this")*
   2. The wide-character designator: *L"like this"*
4. The wxFormBuilder application is newer on Windows than on linux. Therefore, you cannot edit a Windows .fbp file on the linux system.

NOTE: Two commands will help you solve items 1-3 with less work:

* cmpTestForWxWidgets *nameOfFile* – Tests if the file can be compiled. If no errors are reported, you are ready to build your application. **Be sure to test SubGUIClass.cpp.**
* filterWxWidgetsForUnicode *nameOfFile* – Replaces string literals *"like this"* with *L"like this"* and inserts the *wxMessageDialog* include at the top of the file.

**Building Your Program**[[10]](#footnote-10)**:** To build your wxWidgets application, use the following command:

/usr/bin/*g++-4.8 -o HW5 wxWidgetsApp.cpp GUIClass.cpp SubGUIClass.cpp `wx-config --cxxflags --libs`*

Note that *`wx-config –cxxflags –lib`* MUST be APPENDED at the end of the line and that it uses back quote characters (i.e., ` `). Back quote appears to the left of the ‘1’ on most keyboards. Once you’ve successfully built your application, you can run it by typing: *HW5 &*

1. The solution referenced in this document is for Visual Studio 2017. The solution discussed here will work equally well for Visual Studio such as Visual Studio 2015 (however, see footnote 3). [↑](#footnote-ref-1)
2. If you prefer to use the linux environment wxWidgets and wxFormBuilder have been installed on john. See the last page of this document for specific notes for using them on the linux platform. [↑](#footnote-ref-2)
3. Additional instructions for linking VS2015/17 code to wxWidgets appear on the next to last page of this document. [↑](#footnote-ref-3)
4. wxWidgets has already been installed on all CS lab computers. You can confirm installation by verifying the existence of C:\Program Files\wxWidgets2.8. [↑](#footnote-ref-4)
5. If you choose, you can install a full version. However, this choice is only useful if you compile C++ program using the MinGW compiler (e.g, for developers using Eclipse rather than Visual Studio). [↑](#footnote-ref-5)
6. Windows7 and later users may wish to choose “Program Files (x86)” for 32-bit applications. [↑](#footnote-ref-6)
7. These notes have been tested on Windows 7, Windows8 and Windows 10 with Visual Studio 2015/17. However, the preinstalled binaries for wxWidgets DO NOT work. Therefore, please take note of the second-to-last page of these notes: Building a Visual Studio 2015/17 Project w/wxWidgets. [↑](#footnote-ref-7)
8. At one time, wxFormBuilder was limited to generating code into local folders (e.g., the C: drive). It now appears that this limitation is fixed in its latest release. Therefore, if you want to move your project from computer to computer, you may now write it to a USB drive or a network (e.g., H:) drive. [↑](#footnote-ref-8)
9. Link errors use values of 1600 for VS2010 libraries (the default) and 1900 for both VS2015 and VS3017 libraries. [↑](#footnote-ref-9)
10. If you prefer, I have provided a command for building a wxWidgets app called *buildWxWidgetsApp*. The command operates as follows: buildWxWidgetsApp *nameOfExecutable* *listOfSourceOrObjectFilesToBuildFrom* [↑](#footnote-ref-10)