wxWidgets

quick guide to get you started

Contents

- what wxWidgets is
- how to compile it
- how to use it
- examples
- exercises

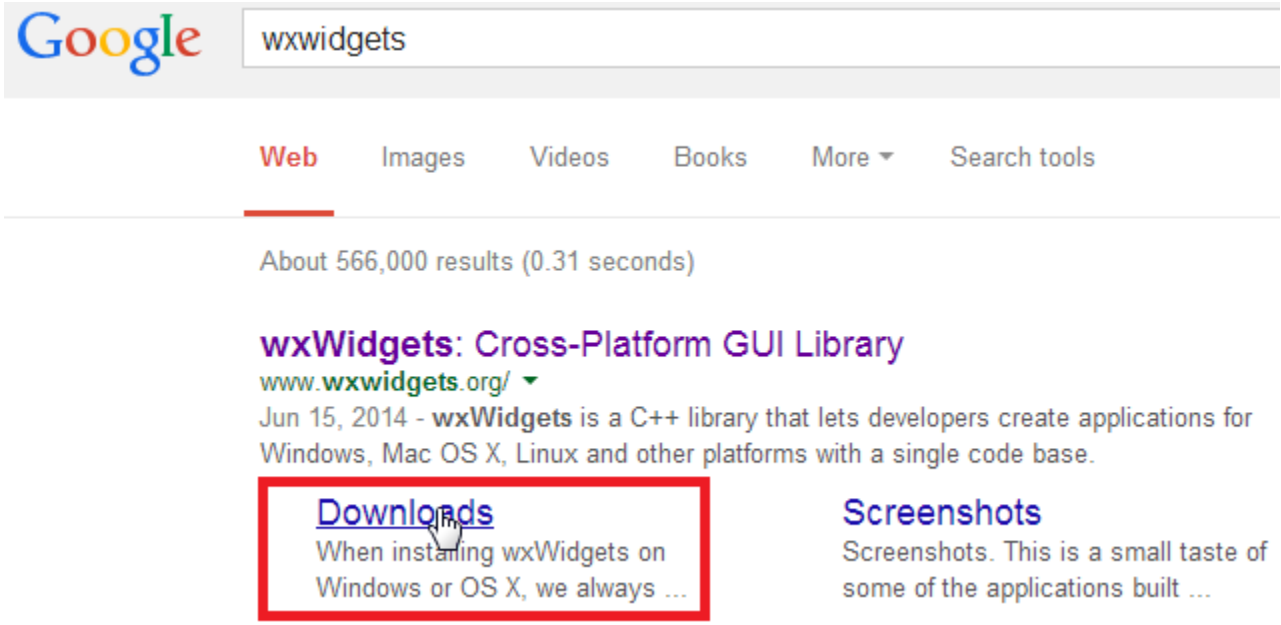
About wxWidgets

- GUI library started in 1992 (*)
- Aims to provide native look on multiple platforms (Windows, Linux, Mac, ...)
- Open Source - wxWidgets license (*)
  - can be used in commercial applications
- Various cross platform modules (**)

(*) http://en.wikipedia.org/wiki/WxWidgets

(**) http://docs.wxwidgets.org/stable/modules.html

Step 1: Get it



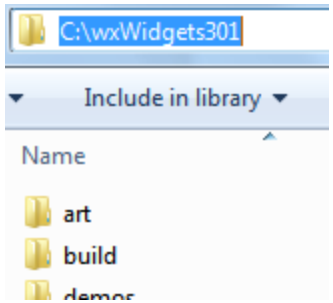
Direct link: http://sourceforge.net/projects/wxwindows/files/

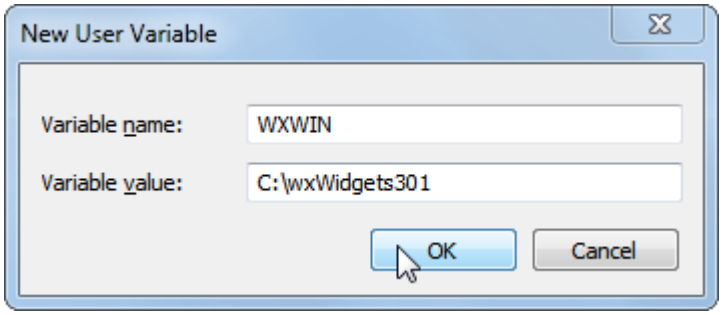
Get:

1. wxWidgets-**X.Y.Z**.zip (The actual library)
2. (optional) wxWidgets-**X.Y.Z**-docs-chm.zip (API documentation)

Step 2: Unzip & setup

- unzip the file to a location without spaces
  - **C:\wxWidgets301** is used here
- set WXWIN environment variable to wxWidgets location (*)

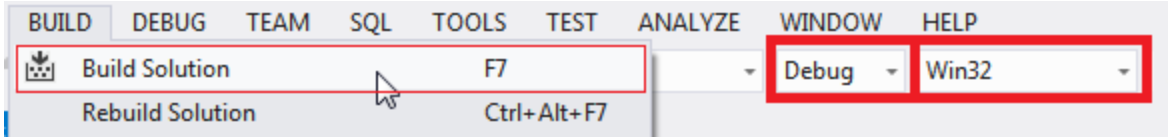




(*) **Control Panel->System->Advanced System Settings->Environment Variables->New**

Step 3: Compile it

- Open solution from $(WXWIN)\build\msw (*)
  - For VS 2010 - open wx_vc10.sln
  - For VS 2012 - open wx_vc11.sln
  - For VS 2013 - open wx_vc12.sln
- For simplicity select Debug/Win32

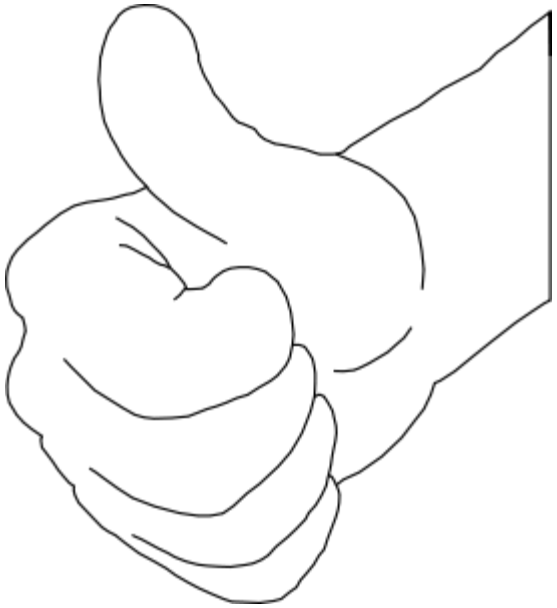


- Build solution

```
23>  wx_vc11_richtext.vcxproj -> C:\wxWidgets301\build\msw\..\..\lib\vc_lib\wxmsw30ud_richtext.lib
========== Build: 23 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

(*) $(WXWIN) is the wxWidgets location (set in the environment variable)

Congratulations!



You have wxWidgets compiled!

Test wxWidgets samples

- open **$(WXWIN)\samples\minimal** project
  - open **minimal_vc9.vcproj**
- compile and run it
- do the same for
  - $(WXWIN)\samples\opengl\cube
  - $(WXWIN)\samples\toolbar
  - $(WXWIN)\samples\ribbon
  - $(WXWIN)\samples\erase
  - $(WXWIN)\samples\dnd
  - $(WXWIN)\samples\docview
  - $(WXWIN)\samples\combo

The event loop

UI processes events similar to this code:

```
while(true) // loop is active until program closes
{
    event = WaitForEvent();
    ProcessEvent(event);
}
```

DO NOT BLOCK processing events, you will block the UI!

Exercise 1

- Analyze the minimal sample program
- See how the events are processed
- Add a new menu command in Help menu
  - Name the command as “Help me”
  - use wxID_HIGHEST + 1 for as new command id
  - Make F2 as shortcut
  - Display a message using wxMessageBox

NOTE: Always check the documentation for functions/classes/macros/examples

Exercise 2

- Open the minimal sample from Exercise 1
- Instead of displaying a message when pressing F2 block the ui for 5-10 seconds
  - use wxThread::Sleep function
- During the sleep try:
  - selecting menus
  - closing the application
- Explain the behaviour

wxWidgets project using cmake (1)

- copy minimal.cpp from minimal sample in a directory (with no spaces)
- create a **CMakeLists.txt** with the following contents:

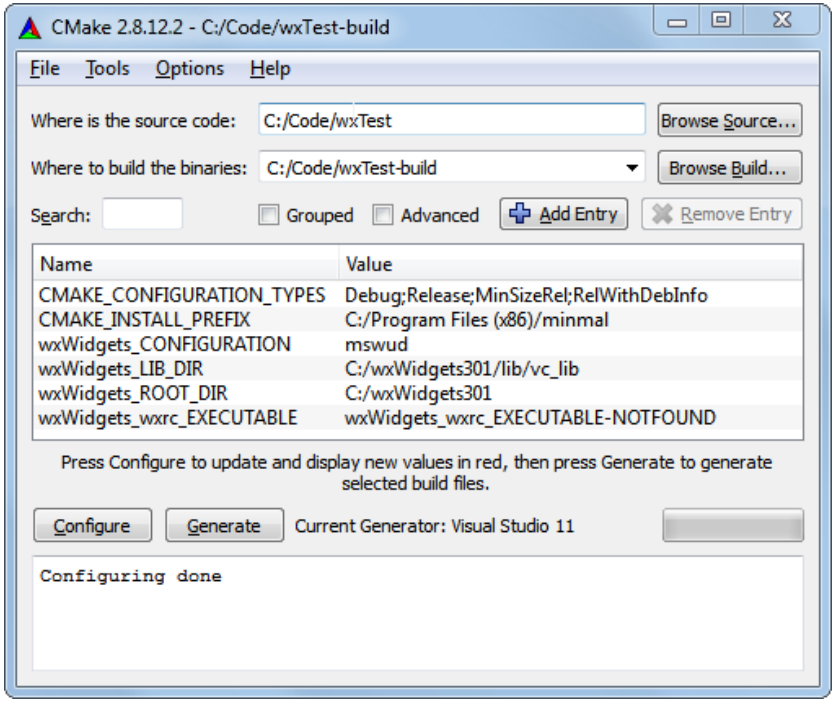
```
CMAKE_MINIMUM_REQUIRED (VERSION 2.6)
PROJECT (minimal)

FIND_PACKAGE(wxWidgets COMPONENTS base core base adv REQUIRED) # other components can be added
INCLUDE(${wxWidgets_USE_FILE})

ADD_EXECUTABLE(minimal WIN32 minimal.cpp)
TARGET_LINK_LIBRARIES(minimal ${wxWidgets_LIBRARIES})
```

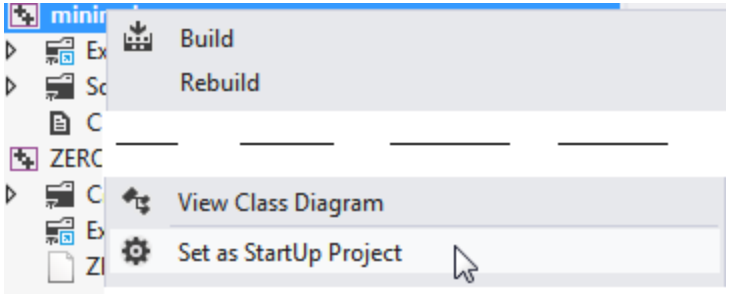
wxWidgets project using cmake (2)

- Start cmake-gui
- Select generator
  - Visual Studio 11 for Visual Studio 2012
  - Visual Studio 12 for Visual Studio 2013
- Generate



wxWidgets project using cmake (3)

- Open minimal.sln from output directory
- Compile the solution
- Set **minimal** as “Startup project”



- Run/Debug the program

Exercise 3

- Add a .cpp & .h file to the cmake project
  - Files should be named “message.h|cpp”
  - They are in the same folder as **minimal.cpp**
  - Declare a **GetTimeAsString** function (in .h file)
    - wxString GetTimeAsString(void);
  - Implement (in .cpp file) the function
    - Should return the value of
      - **wxDateTime::FormatISOCombined**
      - Check **wxDateTime::Now** documentation
- Add an event that displays the result of **GetTimeAsString** function
  - You can use **wxMessageBox** to display message

Creating GUI with a GUI designer

Why:

- Writing GUI by hand is not easy
- It’s not more efficient in most cases

We’ll use wxFormBuilder:

Get it from: http://sourceforge.net/projects/wxformbuilder/

Hint: You can also use the designer to see how the code should be written

Practice with wxFormBuilder

- Start a new project (name it **MyProject**)
- Set the output file name **MyProject**
- Create a new frame (name it **MyFrame**)
- Add a menu bar to the frame
- Create 3 menus (File, Edit, Help)
- Add menu actions (Exit, Open, About, …)
- Add events to each action
- Generate code and analyze it

NOTE: Set the event generation of the project to ***connect.*** You can check to see how ***table*** works also. Do you see some similarities with the minimal sample when using ***table***?

Use the generated code (1)

- Put the generated code into a folder
- Create a main.cpp file with the following contents:

```
#include "MyProject.h"
#include <wx/app.h>
class MyApp : public wxApp
{
public:
   virtual bool OnInit();
};
DECLARE_APP(MyApp)

IMPLEMENT_APP(MyApp)
bool MyApp::OnInit()
{
   if ( !wxApp::OnInit() )
      return false;
   MyFrame *frame = new MyFrame(NULL);
   frame->Show(true);
   return true;
}
```

Use the generated code (2)

- Create a CMakeLists.txt file with all .cpp files:

```
CMAKE_MINIMUM_REQUIRED (VERSION 2.6)
PROJECT (mySample)

FIND_PACKAGE(wxWidgets COMPONENTS base core base adv REQUIRED) # other components can be added
INCLUDE(${wxWidgets_USE_FILE})

ADD_EXECUTABLE(mySample WIN32 main.cpp MyProject.cpp)
TARGET_LINK_LIBRARIES(mySample ${wxWidgets_LIBRARIES})
```

- Follow the steps previously described at “wxWidgets project using cmake”
- compile and run the project

REALLY use the generated code

- From **MyFrame** derive **MyDerivedFrame** (in a different file (.h & .cpp)
- Add the .cpp file to CMakeLists.txt
- Modify main.cpp to instantiate **MyDerivedFrame** instead of **MyFrame**
- Override event methods
- Add your own event handling in the overridden methods
  - a wxMessageBox is enough

NOTE: You have the option in wxFormBuilder to generate code that declares pure virtual methods so that you are **forced** to derive and implement the event handlers

Exercise 4

- Create a frame similar to the following
- Add events for each button
  - *"Start"* disables *"Open"* and *"Save"* buttons
  - *"Stop"* enables them
  - *"Open"* opens a file dialog and sets the title of the window as the chosen file name (see wxFileDialog)
  - *"Save"* opens a save dialog (see wxFileDialog) and displays the selected file in a message box
- Bonus: Add a close event to ask for confirmation at exit (see wxCloseEvent)

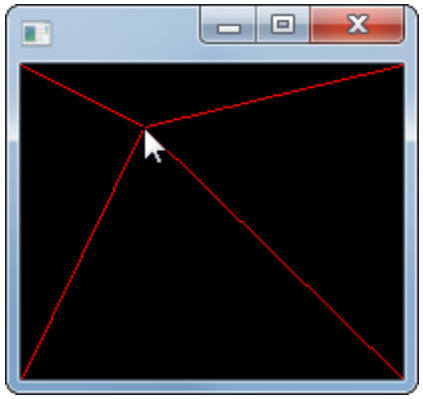
NOTE: To compile follow previously described steps on how to use the generated code

Exercise 5 - Drawing

- Create a frame in wxFormBuilder
- Create a paint event on the frame
- Generate the code and derive from the frame class
- Implement the paint event in the derived class (see wxPaintEvent)
  - Draw vertical lines with red at 10 pixels distance
  - Draw horizontal lines with green at 20 pixels distance
  - Make the background black

Hints: *wxDC::SetBackground*, *wxDC::Clear*, *wxDC::GetSize*, *wxDC::SetPen, wxDC::DrawLine*

Exercise 6 - Mouse Events



- Get the code from Exercise 5
- Add a mouse movement event to the frame (See *wxMouseEvent* and *wxWindow::Refresh*)
- Modify the paint event to draw 4 lines connecting the current mouse position with the 4 corners of the panel (see *wxGetMousePosition* function & *wxWindow::ScreenToClient*)
- Lines should be red; background should be black

BONUS:

- Draw directly from the mouse movement handler (see *wxClientDC*)

Solutions to exercises

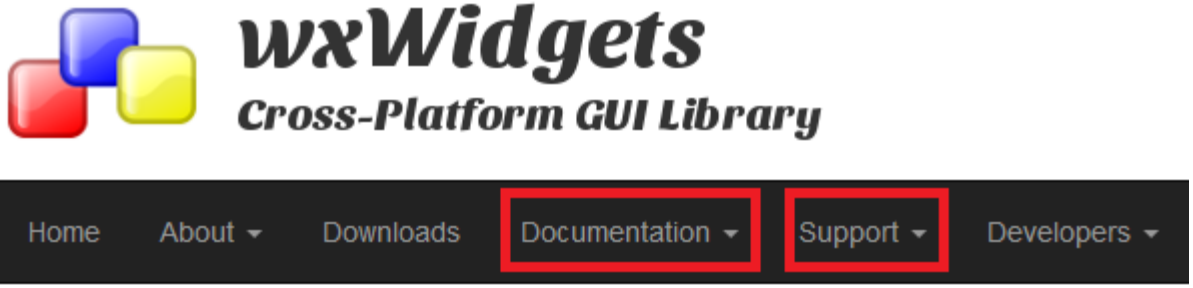
Mercurial(Hg) repository:

https://bitbucket.org/undergraver/wxwidgetsintro

NOTE: Update gradually to see the actual difference between commits

I’m stuck, I need help

- Resources from http://www.wxwidgets.org/



- Or use http://stackoverflow.com/
  - tag accordingly!