SUNY Geneseo – June 15, 2010

GTk+ and GTkGLExt Build Process for Windows 32-bit

Using Minimal GNU for Windows (MinGW) and Minimal System (MSYS)

Author : Hieu Quang Tran

Advisor : Professor Doug Baldwin

This is a guide to building GTk+ and GTkGLExt in Microsoft Windows 32-bit. GTk+ is a toolkit for creating graphical user interfaces, and GTkGLExt is an OpenGL extension to GTk+ that provides support for OpenGL rendering. This guide will instruct you how to build GTk+, GTkGLExt and all of their dependencies.

Contents

Pre-build Installations

Before you start building GTk+, you need to install MinGW and MSYS. These two pieces of software will create a Unix-like terminal where you can type Unix commands to build libraries of GTk+ and all of its dependencies from source code.

Install Minimal GNU for Windows (MinGW)

1. The official website of MinGW (www.mingw.org) provides free copies of the software. The easiest way to get MinGW is to download the automated installer, which is an executable file that will download all the individual packages and install them into your computer.
2. "The ***default*** installation directory, as pre-configured within the installer, is *C:\MinGW*. Unless you are ***very*** sure you know what you are doing, ***you should not change this***; in particular: ***never install MinGW into any directory which includes spaces in its absolute path name***." (source: www.mingw.org)

Install Minimal System (MSYS)

1. An automated installer of MSYS can also be downloaded from the official website of MinGW (www.mingw.org). Usually, MSYS is installed into C:/msys/1.0, but you can install it anywhere you prefer.
2. **Post-Installation Process**: After the installation has finished, a terminal will appear and ask if you want to perform the post-installation process. Please type "y" for yes.

   The process will then ask if you have already installed MinGW. It is strongly recommended that MinGW is installed before MSYS. Please type "y" for yes if you have done so.

   Next, you will be asked for the path name of the directory where MinGW was installed. Please enter "c:/mingw" if this is where you have installed MinGW.

3. Now you should have a cyan "M" link on the Desktop. When you double-click on it, a terminal should be launched.
4. If MSYS is installed into C:/msys/1.0, then C:\msys\1.0\local will be mapped [*in MSYS's own "perspective," i.e., interpretation of the Windows file system*] as: /usr/local. However, **installing packages to "/usr/local" should be avoided, since the MinGW compiler won't look there by default for, e.g., library or header files. It is strongly recommended that any package be installed into /mingw [in MSYS own perspective], or in other words, C:/MinGW**
5. For more information, please visit MinGW official website at www.mingw.org

Install Minimal System Development Toolkit (MSYS-DTK)

MSYS-DTK is a package that provides MSYS with many add-ons, functionalities, and packages, such as autoconf, automake, libtool, autogen, openssl, openssh, cvs, guile, and inetutils. It also provides MSYS with perl, which is sometimes required to build some programs.

Please click on the link below to download MSYS-DTK version 1.0.1 (The latest version, as of this writing) from source-forge:

http://sourceforge.net/projects/mingw/files/MSYS/Supplementary%20Tools/msysDTK-1.0.1/msysDTK-1.0.1.exe/download

The file that you have downloaded is an automated installer. Please run this executable file to install the package into the directory where you have installed MSYS. For example, if you have installed MSYS into C:\msys\1.0, then this is the place where you should put MSYS-DTK.

Build Shared Libraries of GTk+

Before you can actually start building GTk+, you need to build all of its dependencies, including Glib, Cairo, Pango, and ATK. To build each of these packages, again, you need to build all of its dependencies. For example, Glib requires Zlib, LibIconv, Pkg-Config, and Gettext; Cairo requires Zlib, Glib, LibPNG, and Pixman. When you get to building Pango and ATK, you will have already built most of the packages they require.

Please keep in mind that in order to build either shared or static libraries of GTk+, you must have shared libraries of all of its dependencies. The reason is that GTk+ needs to link against its dependencies, and in order to do that it needs those packages to be shared.

Generally, the process of building a package includes three stages:

1. **Download the latest version of the package**: All of these packages are free and can be downloaded from their home websites. The recommended download locations of each of these packages will be mentioned later when we go into details.
2. **Uncompress the package**: Most of the time, the packages downloaded will be compressed in the form of a tar- or gz- file. You need to have an uncompress program to uncompress them. I would recommend using a piece of software called 7-zip. It is an open source Windows utility for manipulating archives. It is totally free and can be downloaded from its home website: http://www.7-zip.org/
3. **Build the package**: Finally you can start building all the packages that you have downloaded. There are a few steps in building them:
    a. **Open the MinGW terminal** that you have installed before. One way to do that is to double click on the cyan "M" icon on your computer desktop. Also, you can go to your Start Menu → All Programs, and select MSYS folder; then select "MSYS". The terminal will be activated.
    b. **Move the current working directory to the directory of the uncompressed package that you want to build**. Having opened the MinGW terminal, now you want to type in the terminal a command that has syntax like **cd [path name to the directory of the package]**. For example, if you want to build zlib, and you have put an uncompressed copy of your zlib in the directory c:/zlib, then you want to type in the terminal: **cd c:/zlib**.
    c. **Type build commands**. The actual build process consists of a sequence of commands that you need to type into the terminal. These commands will be given later when we go into details. Just keep in

mind that each of these commands will give your computer a task to perform, and it takes time to finish that task. Also, it is important that you type these commands in the given order. **One last thing is that these commands will build your packages into c:/mingw. If you want to build your packages to some other place, just change the path name. However, it is not recommended that you do so, since MinGW, by default, will look into only c:/minGW to find all the dependencies that your package needs.**

d. **Note that many build commands include options beginning with hyphens (e.g., "—prefix…"). Some of these options begin with one hyphen, some with two. Pairs of hyphens may appear as a single long dash in the examples below.**

Build Glib

The first dependency of GTk+ is Glib. In order to build Glib, you need to build the following packages first: Zlib, LibIconv, Gettext, and Pkg-Config. Glib requires these packages to be built as shared libraries so that it can link to them. For most of the packages, building shared libraries is a default option. Therefore, you do not need to explicitly enable shared-libraries option.

Build Zlib

1. The latest version of Zlib is 1.2.5, as of this writing. You can download it from its home website at http://www.zlib.net/.
2. **Build commands**. After moving your current working directory to the directory of the uncompressed zlib, type the following commands into the terminal in the given order:

```
cp win32/makefile.gcc makefile.gcc
make -fmakefile.gcc
export "INCLUDE_PATH=/mingw/include"
export "LIBRARY_PATH=/mingw/lib"
```

make install -fmakefile.gcc

cp zlib1.dll /mingw/bin

(Source: http://www.gaia-gis.it/spatialite-2.4.0/mingw_how_to.html)

3. There is a bug in the build of this version of Zlib, and that is the name of the dynamic-link-library-index file. The file is produced as the result of the build process and is put in the folder of the zlib package. However, its name is libzdll.a, not libz.dll.a, which is how it is supposed to be (a dot is missing between "libz" and "dll"). Therefore, you need to rename the file first and then copy it to the library folder in the directory where you choose to build your library. To do that, type the following commands into the terminal:

   mv libzdll.a libz.dll.a

   cp libz.dll.a c:/mingw/lib

Build LibIconv

1. The latest version of LibIconv is 1.13.1, as of this writing. You can download it from one of the following web pages:
   a. Libiconv homepage: http://www.gnu.org/software/libiconv/
   b. All versions of Libiconv: http://ftp.gnu.org/pub/gnu/libiconv/
2. **Build commands**: After moving your current working directory to the directory of the uncompressed libiconv, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

   make

   make install

Build Gettext

1. The latest version of Gettext is 0.18, as of this writing. You can download it from one of the following web pages:
   a. Gettext homepage: http://www.gnu.org/software/gettext/
   b. All versions of gettext: ftp://mirrors.kernel.org/gnu/gettext/

2. **Build commands**: After moving your current working directory to the directory of the uncompressed gettext, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

   make

   make install

Build Pkg-Config

At this moment, you cannot build pkg-config from its source code because of a circular dependency that pkg-config and Glib have together. The best thing to do right now is to just use a copy of a statically pre-built version of pkg-config so that you can break this circular dependency, and then you can rebuild Pkg-Config later on.

To download a copy of the executable file pkg-config.exe, please click on this direct link: http://cfile228.uf.daum.net/attach/161B090C4A84DF106B0645

After having downloaded the executable file pkg-config.exe, you want to put that file into the 'bin' folder, which is under the directory where you build all your libraries. For example, if you are building your libraries at c:/mingw, then the path name of the bin folder is c:/mingw/bin

Build Glib

After having built all of its dependencies, now you can start building Glib. As mentioned above, Glib needs to be built as a shared library so that GTk+ can link to it. No special action is required since building shared libraries is already a default option of Glib configure file.

1. The latest version of Glib is 2.25.7, as of this writing. You can download it from one of the following web pages:
   a. Glib source code repository: http://git.gnome.org/browse/glib/
   b. All versions of glib: http://ftp.gnome.org/pub/gnome/sources/glib/
2. Build commands: After moving your current working directory to the directory of the uncompressed Glib, type the following commands into the terminal in the given order:

./configure –prefix=/mingw

make

make install

Build Cairo

The second dependency of GTk+ is Cairo. Cairo requires you to build the following additional packages: Pkg-Config, LibPNG, and Pixman.

Build Pkg-Config

Even though you have already included a statically prebuilt version of Pkg-Config into the library, it is just to get you through building Glib. Now, you need to actually build Pkg-Config to continue building other packages.

1. The latest version of Pkg-Config is 0.25, as of this writing. You can download it from the following repository: http://pkg-config.freedesktop.org/releases/
2. Build commands: After moving your current working directory to the directory of the uncompressed Pkg-Config, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

   make

   make install

Build LibPNG

LibPNG is required by Cairo since Cairo uses some of its functions. Unfortunately**, there is currently a bug in building LibPNG**, and it is impossible to build it in a way that will work with Cairo. This issue is discussed on this webpage: http://stackoverflow.com/questions/2151145/building-cairo-for-windows-with-mingw-problems-linking-libpng.

The solution to this right now is to just download an already-built copy of LibPNG and put it into the library. To download a copy of pre-built LibPNG, please go to this webpage: http://www.gtk.org/download-windows.html

On this webpage, if you scroll down to the place where the header line is Third Party Dependencies, you will see a list of packages, versions and download links. Please download both the Binaries and Development (Dev) of the package libpng. The current version of the LibPNG package on this webpage is 1.4.0, as of this writing. **Please Do Not Uncompress Them Yet!** After having downloaded these two compressed files, you want to copy them into the folder where you build your libraries. For example, if you build your libraries in the directory c:/mingw, then you want to put these two compressed files in that directory. Now, you uncompress them. The files of the LibPNG library will be automatically put into the right locations.

Build Pixman

1. The latest version of Pixman is 0.18.2, as of this writing. You can download it from the following repository: http://cairographics.org/releases/
2. Build commands: After moving your current working directory to the directory of the uncompressed Pixman, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

   make

   make install

Build Cairo

1. Finally, you can build Cairo. The latest version of Cairo is 1.8.10, as of this writing. You can download it from the following repository: http://cairographics.org/releases/
2. Build commands: After moving your current working directory to the directory of the uncompressed Cairo, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

make

make install

Build Pango

Pango is the third dependency of GTk+. Like Glib and Cairo, Pango also depends on a lot of other packages. Fortunately, you have built most of them all already. If you just want to build GTk+, but not GTkGLExt, you can actually start building Pango right now. However, that will lead to some lack of components; specifically the library pangoft2 will not be built. This library is not required to build GTk+, but it is needed for GTkGLExt to be properly built. Therefore, it is a good practice to build Pango properly. To do that, before start building Pango, you need to build FreeType and FontConfig. FontConfig requires FreeType and another package, which is Expat. Therefore, you need to build Expat and FreeType before building FontConfig. The suggested order of packages to be built is Expat, FreeType, FontConfig, and Pango.

Build Expat

1. The latest version of Expat is 2.0.1, as of this writing. You can download it from the following webpage: http://www.sfr-fresh.com/unix/www/expat-2.0.1.tar.gz/
2. Build commands: After moving your current working directory to the directory of the uncompressed Expat, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw –disable-static –enable-shared

   make

   make install

Build FreeType

1. The latest version of FreeType is 2.3.12, as of this writing. You can download it from the following repository: http://download.savannah.gnu.org/releases-noredirect/freetype/

2. Build commands: After moving your current working directory to the directory of the uncompressed FreeType, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw –disable-static –enable-shared

   make

   make install

Build FontConfig

1. The latest version of FontConfig is 2.8.0, as of this writing. You can download it from the following repository: http://www.fontconfig.org/release/
2. Build commands: After moving your current working directory to the directory of the uncompressed FontConfig, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw –disable-static –enable-shared

   make

   make install

Build Pango

1. The latest version of Pango is 1.28, as of this writing. You can download it from the following repository: http://ftp.gnome.org/pub/GNOME/sources/pango/
2. Build commands: After moving your current working directory to the directory of the uncompressed Pango, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

   make

   make install

Build ATK

All the dependencies of ATK have already been built, so you can just starting build ATK right away.

1. The latest version of ATK is 1.30, as of this writing. You can download it from the following repository: http://ftp.gnome.org/pub/gnome/sources/atk/
2. Build commands: After moving your current working directory to the directory of the uncompressed ATK, type the following commands into the terminal in the given order:

   ./configure –prefix=/mingw

   make

   make install

Build GTk+

Congratulations! You have built all the dependencies of GTk+. The final step is to build GTk+ itself. The GTk+ version that I have built is 2.16.6, which is not the newest one, as of this writing. If you try to build some later versions of GTk+, the build process might be different depending on the version the GTk+ package you are trying to build. Building GTk+ requires some extra work besides just typing commands into the terminal, so let us get started:

1. **Configure**: There are two problems that you need to handle before start the configure process:
   a. MSYS has a bug that makes an illegal character (apparently control-A, ASCII code 0x01) appear in the configure file of GTk+. When reading that character, MSYS cannot understand it, so it represents it as a square. If you start the configure process right now, that character will cause the gcc compiler to throw an "Invalid argument" message, making configure believe that certain files (e.g., cairo-pdf.h) are missing, even though they are clearly in the include folder. To get around this bug, here is what you need to do: go to the folder where you put the uncompressed GTk+ package, find the configure file (the name of the file is "configure" without any extension) and open it with a text-

editor. You want to search the entire document for "GTK_DEP_CFLAGS". This is an environment variable. There are many places where this variable appears. You want to go to place where it is defined as something similar to (**Note**: *The quoted codes below are supposed to be on a single line in the file named "configure". In this document, Word wraps them onto multiple lines*):

```
GTK_DEP_CFLAGS="`$PKG_CONFIG --cflags  gthread-2.0
$GDK_PIXBUF_PACKAGES $GDK_PACKAGES $GTK_PACKAGES`
$GDK_PIXBUF_EXTRA_CFLAGS $GDK_EXTRA_CFLAGS $GTK_EXTRA_CFLAGS"
```

You want to delete the last three variables from the definition of GTK_DEP_CFLAGS. After doing that, you will have something like this:

```
GTK_DEP_CFLAGS="`$PKG_CONFIG --cflags  gthread-2.0
$GDK_PIXBUF_PACKAGES $GDK_PACKAGES $GTK_PACKAGES`"
```

Please remember to save the change that you have made.

b. The second problem comes from the conflicting naming conventions between LibPNG and GTk+. Please keep in mind, **this problem only occurs if the LibPNG package you have built is not version 1.2. If you have a LibPNG package version 1.2, then please do not bother to read this section since GTk+ configure file has a separate process to check for libpng12.dll.a**. If your LibPNG is a version later than 1.2, then the build process cannot find libpng12.dll.a; it will, then, go checking for a general LibPNG library, and you will get a problem here. For example, in my case I have LibPNG version 1.4.0; the index file of the dynamic link library of LibPNG in the lib folder should have the name libpng14.dll.a, which suggests that the name of the library is libpng14. However, this is not how GTk+ refers to LibPNG; it refers to LibPNG as libpng. **Therefore, when GTk+ goes to look for LibPNG, it looks for the file libpng.dll.a, not libpng14.dll.a**. As a result of that, GTk+ cannot find LibPNG and throws error messages. To fix this, you want to create a file with the name "libpng.dll.a" in the same directory with the file libpng14.dll.a, and, at the same time, you need to link this newly created file with the file libpng14.dll.a. Here are the steps of how to do that:

    - Open the MinGW/MSYS terminal.

- Move your current working directory to the folder named “lib” in the directory where all of your libraries are built. For example, if I build all of the libraries into the directory MinGW with the path name C:/MinGW, I should move the current working directory to the directory named “lib” inside that MinGW folder by typing into the terminal:

  cd c:/mingw/lib

- Type the following command next:

  ln –s libpng14.dll.a libpng.dll.a

Through doing this, you will create a file named libpng.dll.a in the same directory with and linked to the file libpng14.dll.a. Thereby, when the build process of GTk+ goes looking for the file libpng.dll.a, it will be forwarded to the file libpng14.dll.a, which is the correct file.

Having done all of this, now you can start the configure process. Open the terminal, move your directory to the directory of the uncompressed GTk+ package and type the following command:

./configure –prefix=/mingw –without-libjasper –without-libjpeg –without-libtiff

As you can see in the command line above, I have disabled linking GTk+ to a lot of packages. The reason is that they are not required by GTk+; they are just optional packages, and if you want to link GTk+ to these packages, you must have already built them. For example, if you enable option to link GTk+ to libjasper, the build process will throw an error of not finding libjasper library. LibPNG is different, even though it is an optional feature of GTk+, it is required by Cairo, so I decided to include it as part of the GTk+ libraries. If you want GTk+ to have any of the features above, just build the corresponding package that GTk+ requires before building GTk+.

2. **Make**: As mentioned above, MSYS has a trouble with interpreting a character in GTk+ file “configure”. If you have followed the steps above, you should be able to get around that bug. However, the configure process produces a make-file, and this

undefined character follows the configure process to the produced make-file. The problem is that there are tons of make-files produced by GTk+ configure. Therefore, this undefined character is rampant. To fix this, you need to write a shell script that will find all the appearances of this character in all the make-files in the GTk+ package folder and delete them. Here are the steps:

a. **Create a text file:** In this step, you need to create a text script file and put it in an appropriate location. In order to do that, first, you want to create a blank text file (a file with the extension .txt). There are a lot of ways to do that; one of them is to simply right click on the blank space on the window; a menu will pop up. You move your cursor to "New"; another menu will appear; on this menu, you select the option called "Text Document". This will give you a text file with the name "New Text Document.txt". Then, you want to rename this file to "trd001.txt". Finally, you need to put this text file inside a directory named "bin". You can put this "bin" directory anywhere you want, but please remember the path name to that directory.

b. **Add contents to the file trd001.txt:** Please open the blank file trd001.txt that you have created with a text editor. Please delete all the current contents of that file and insert the following lines of code:

```
#!/bin/sh
tr -d '\001' < $1 > /tmp/temp.txt
cp /tmp/temp.txt $1
```

c. **Enter commands to the terminal:** Please open the MinGW/MSYS terminal and do the following steps:

- Move the current directory to the directory of your uncompressed GTk+ package. For example, if I have put my uncompressed GTk+ package at c:/gtk, then I would enter the following command into the terminal: **`cd c:/gtk`**
- When you are in the directory of your GTk+ package, enter the following commands in the given order:

mv c:/gtk/bin/trd001.txt c:/gtk/bin/trd001.sh

chmod a+x c:/gtk/bin/trd001.sh

find -iname Makefile -exec c:/gtk/bin/trd001.sh \{\} \;

These commands are typed based on the assumption that the original file trd001.txt is saved in a folder named "bin", and the location of that "bin" folder is c:/gtk/bin. If you have put that "bin" folder somewhere else, please change the commands accordingly. It will take MSYS a while to find all the invalid characters and delete them.

Information Source: http://kemovitra.blogspot.com/2009/07/mingw-hack-quickly-removing-0x01-from.html

After having done all of the steps above, now you can start the make process normally by entering **make** into the MinGW/MSYS terminal.

3. **Make Install:** If you get to here, it means that you have got through all the hard parts. Now, all you need to do is to start the make-install process as usual. Just type **make install** into the MinGW/MSYS terminal and let it run.

Build Shared Libraries of GTkGLExt

In order to perform OpenGL tasks on windows provided by GTk+ you need to install GTkGLExt. You have already built all the dependencies of GTkGLExt. There is one problem, however, that needs to be handled before GTkGLExt can actually be built. That problem is in the build process, GTkGLExt requires a library called pangox. This library is the X–Windows backend of the Pango library. It is totally normal not to have this library on a Win32 installation. You need to hack into the build process of GTkGLExt to remove this requirement. To do that, you just simply need to remove all lines that involve the world "pangox" in the two files configure (the name of the file is "configure" without any extension) and configure.in. Please keep in mind that the version of GTkGLExt being built is 1.2.0, which is the latest version, as of this writing. Different versions of GTkGLExt may require different build steps.

1. **In the file configure**: You need to make the following changes:
   a. Remove all the lines similar to:
      `pangox >= 1.0.0 \\`
      and

```
pangox >= 1.0.0 \
```

b. Remove the word "pangox" from the line:

```
GDKGLEXT_PACKAGES="gdk-2.0 pango pangox gmodule-2.0"
```

2. **In the file configure.in**: You need to make the following changes:

a. Remove this entire block of codes:

```
# Pangox
m4_define([pangox_pkg], [pangox])
m4_define([pangox_required_version], [1.0.0])
```

b. Remove this line of codes:

```
pangox_pkg >= pangox_required_version \
```

c. Remove the phase "pangox_pkg" from this line of codes:

```
GDKGLEXT_PACKAGES="gdk_pkg pango_pkg pangox_pkg
gmodule_pkg"
```

Once, you are done with all the steps above, it is time to starting building GTkGLExt. The latest version of GTkGLExt is 1.2.0, as of this writing. You can download GTkGLExt from the following webpage: http://projects.gnome.org/gtkglext/download.html

After moving your current working directory to the directory of the uncompressed GTkGLExt, type the following commands into the terminal in the given order:

./configure –prefix=/mingw –disable-static –enable-shared

make

make install

Build Static Libraries of GLib

It appears that GTk+ cannot be built statically on Windows. If you try to build GTk+ statically by disabling the shared option and enabling the static option in the configure stage like this: **`./configure –prefix=/mingw –disable-shared –enable-static`**, then the configure will give this back to you:

Because of that reason, building GTk+ statically will not be mentioned in this write-up. Instead, let us talk about how to build static libraries of GLib. Please keep in mind that **in order to build GTk+, you must have shared libraries of Glib**. Therefore, building Glib statically is not recommended if your goal is to build GTk+. However, you can always have both shared and static libraries of Glib. To have that, you just need to build Glib twice, first time you build shared libraries, and second time you build static ones. However, just keep in mind that you cannot build shared and static libraries of Glib at the same time. You must use separate build processes.

In order to build Glib, regardless of shared or static, all of its dependencies must be built dynamically so that Glib can link to them. Therefore, you must fist build shared libraries of all of its dependencies, including ZLib, LibIconv, Gettext, and Pkg-Config. Please just follow exactly the same steps explained in building shared Glib. After that you just need to make some small changes to the build process of Glib. These changes are made in the commands that you type into the terminal to build your packages:

Build Glib Statically

For each Glib build process, you cannot build both a shared and a static library of Glib at the same time on Windows. In fact, if you type into the terminal this command **./configure –prefix=/mingw –enable-static**, an error will be thrown with the message "Cannot build both shared and static at the same time on Windows."

Therefore, to build Glib statically, you need to disable share-option first and then enable static-option:

```
./configure –prefix=/mingw –disable-shared –enable-static
make
make install
```

Using OpenGL, GTk+, and GTkGLExt

This section talks about how to link your project with OpenGL, shared GTk+ and GTkGLExt libraries when you create a project using GTk+ and OpenGL functions. The Integrated Development Environment that I used was Code::Blocks, version 8.0.2.

To link your project with the Opengl, GTk+ and GTkGLExt libraries, go to Project on the menu bar → Build Options. A small window will pop up. To the left of the window, you will see your project tree. Please make sure that the name of your project is highlighted, not "Debug" or "Release". These are the next things that you need to do:

1. Click on the Linker Settings tab, which is on the right of the window. Under Link libraries, you want to add the following libraries (**Note**: This list gathers libraries into groups for expository purposes, do **not** enter the group titles into Code::Blocks):
    a. **OpenGL libraries:**
        opengl32
        glu32
    b. **Glib libraries:**
        libglib-2.0
        libgthread-2.0
        libgobject-2.0
    c. **GTk+ libraries:**
        libgtk-win32-2.0
        libgdk-win32-2.0
        libgdk_pixbuf-2.0
        libpng14
    d. **GTkGLExt libraries:**
        gtkglext-win32-1.0
        gdkglext-win32-1.0

2. Click on the Search Directories tab, there are smaller sub-tabs inside this tab.
    a. Click on the Compiler sub-tab and add the following path-names:

- **General header-file folders:**

  C:\MinGW\include
- **Glib header–file folders:**

  C:\MinGW\include\glib-2.0

  C:\MinGW\include\glib-2.0\glib

  C:\MinGW\include\glib-2.0\gobject

  C:\MinGW\lib\glib-2.0\include
- **GTk+ and its dependencies header-file folders:**

  C:\MinGW\include\gtk-2.0

  C:\MinGW\lib\gtk-2.0\include

  C:\MinGW\include\cairo

  C:\MinGW\include\pango-1.0

  C:\MinGW\include\atk-1.0

  C:\MinGW\include\libpng14
- **GTkGLExt header-file folders:**

  C:\MinGW\include\gtkglext-1.0

  C:\MinGW\lib\gtkglext-1.0\include

b. Click on the Linker sub-tab and add the following path-names:

   C:\MinGW\lib

   C:\MinGW\bin

3. Click on the Compiler Settings tab. There are smaller sub-tabs inside this tab. Please click on the Other Options tab and type this command into the field:

   -mms-bitfields

All the above specifications are made based on the assumption that GTk+ and GTkGLExt libraries are built in the directory C:\MinGW. If you have built your library somewhere else, please just change the initial path name.

Using Static Glib Library

If you are using a static Glib library, **you still need to link your project to the Glib library like we have done above with the shared Glib library**. However, since Glib is built statically, you might encounter the following errors when building your project:

1. Undefined reference to "_libintl_fprintf"
   To solve this, you also go to Project on the menu bar → Build Options. In the Build Options window, you click on the Linker Settings tab. Under Link libraries, you want to add a library named "libintl".
2. Undefined reference to "_CoTaskMemFree@4"
   To solve this, get to the Linker Settings tab in the Build Options window. Under Linker libraries, you want to add a library named "libole32".