Dr. E.Huckert 08-2024, 1-2024

How To Build HuNoEd V2

Prequisites

Skills

HUNoEd is a rather complicated program. If you want to modify it or to build it from scratch you should have:

1. a good knowledge of classical music theory
2. experience in object-oriented C++ programming
3. a good understanding of concurrent programming (threads)
4. experience with classical make files
5. experience with wxWidgets

Installation and Packages

- WxWidgets must be installed, version 3.2.2 preferred
- GNU g++ compiler and related utilities, g++ from Mingw preferred
- Additional MIDI software (Linux only: qsynth and soundfonts)
- Font "Noto Music" must be installed

I used wxWidgets in versions 3.1.0 and 3.2.2 to build the software. Other wxWidget versions were not tested.

wxWidgets can/must be configured correctly to build this software. **Static** linking is preferred.

Here some wxWidgets details as obtained by command „wx-config“:

default **config**:

Default config is gtk2-unicode-static-3.2

available **libraries:**

xrc stc richtext ribbon propgrid aui html qa adv core xml net base

We use normal the **normal make** system – **not cmake**!

Under Windows the standard **make utility** may be called mingw32-make or similar. I used the GNU g++ compiler from **Mingw** with option -m32, i.e. I compiled and linked for a 32-bit executable. The (newer) GNU compiler from the cygwin/TCC variant made problems.

There are two make-procedures - depending on the target operating system:

- **Windows 11**: file noedw.mak. Under Windows 11 I have used wxWidgets V3.1.0. The program (the executable) runs also under Windows 7 and Windows 10 and Windows 11
- **Linux/Ubuntu**: file noedu.mak. Under Linux/Ubuntu I have used wxWidgets V3.2.2. The GTK libraries (gtk2-unicode) are the GUI base. We didn't test the program on other Linux distributions.

Before starting the make procedures the pathes must be controlled and changed. It is very probable that my standard make files don't work for everyone. The wxWidgets utility **wx-config** may help to find the correct pathes and options.

The build/make process

Everything happens in the directory where the ZIP has been unpacked. The zips are called:

- hunoeduV2.zip for Linux
- hunoedwV2.zip for Windows

Both zips contain executables produced by me. These executables will be overwritten by the build process.

To **start the build process**

The "x" in the names below must be replaced by "u" for Linux and by "w" for Windows.

1. Unzip the zip in a completely empty directory

```
unzip hunoeduV2x
```

2. If the wxWidgets version has been correctly installed and if utility wx-config works then everything should work. If not you may have to edit or replace these lines in the make file *noedx.mak*:

```
P=c:\users\huckert\wxWidgets-3.1.0
I=$(P)\include
L=$(P)\lib\gcc_lib
COMP=c:\mingw\bin\g++
```

These lines define where the wxWidgets installation directory is (here ...\huckert\wxWidgets-3.1.0) and which compiler should be used (here ...\mingw\bin\g++)

3. If necessary (if *make* is not started for the first time): remove old make results:

```
make -f noedux clean
```

4. Start the compile/link steps:

```
make -f noedx.mak
```

The results should be executables name *hunoed.exe* (Windows) or *hunoedu* (Linux/Ubuntu) in the build directory.

Remarks for the build process

We use UNICODE in both operating systems as the standard encoding for characters and strings. UNICODE uses (for european languages) a 8-bit encoding with special multibyte enhancements for less frequent characters. Note that this says nothing about the internal encodings used by the operation systems (Windows: 2-byte, Linux 4-byte).

For both operating systems we rely on the GNU g++ compilers. The application is **linked statically.** I did not test the make procedures for dynamic linking! Note that wxWidgets must have been built and installed for static linking!

Under Windows we build a 32-bit executable – we did not test the 64-bit mode!

Note that for Windows include file „wx.h“ must be included before the standard Windows includes (else problems with types like LPCSTR etc. - probably related to the UNICODE encoding that we use).

Other prerequisites

1. If the font "Noto Music" is not installed take the font file "NotoMusic-Regular.ttf " and install the font in the operating system. Misplaced or lacking glyphs for musical symbols like segno, coda etc. indicate a missing font.

2. If necessary modify the initialisation file *noed.ini*. This is a conventional key/value based ASCII file as known from the MSDOS/Windows world. The *xdim* and *ydim* entries are important for the view, i.e. the visible extract of a score. Other important entries are *linedist* and *systemdist* which define the visual appearance of the score on the screen and in the printouts. The extry *maxmeasperline* restricts the number of measure per visible line. Many of these entries can be modified at run time via the menus "Settings/Settings (active)" and "File/New",

3. MIDI player : Under Linux you should install and start qsynth as backend for the buildin-MIDI player. Timidity instead of qsynth is also supported but produced in my environment a lot of problems especially when *Pulseaudio* was running. For Windows no additional effort is needed as the MIDI player is included in the operating system. It may be necessary to modify some MIDI related entries in section [MIDI] in file *noed.ini.*

Restrictions

I did **not** test other Linux distributions (Fedora, SuSe etc.) and other GUI packages wxWidgets with GTK only)