XCSoar – Änderungen am Source

**bool small** – umgewandelt in **bool is\_small**, ‚**small** ist in MSVC eine Typangabe

**near** – umgewandelt in **bool is\_near**, ‚**near** ist in MSVC eine Typangabe

**TriState::TRUE** usw. - Umgewandelt in **TriState::True**, damit umgeht man das Macro-Problem (ansonsten ist in MSVC das normale Windows-Result z.T. ausgeblendet!

**#include „Util/Compiler.hpp“** - in der Regel ganz nach oben verlegt (also als erstes include) sonst gibt es Probleme mit WinSock2 vs. WinSock!

Für die Einbeziehung des Script-Paketes LUA muss ich wahrscheinlich noch etwas machen bisher nur den (externen!) Include-Pfad herangezogen!

n

Boost.Asio

XCSoar benötigt boost\_asio, doch das ist keine eigenständige lib, sondern wird ‚anders‘ in das Project eingebunden

Benötigt:

* Boost\_system
* Boost\_thread
* Boost\_date\_time
* Boost\_regex
* Boost\_serialization

Außerdem soll in EIN File

* #include <boost/asio/impl/src.hpp>

Eingebunden werden und BOOST\_ASIO\_SEPARATE\_COMPILATION als Project-Option definiert werden!

If using Boost.Asio's SSL support, you will also need to add

* #include <boost/asio/ssl/impl/src.hpp>.

## Änderungen

Windows / MSVC-Compilierung

* Ursprünglich war es für MSVC kompiliert, immerhin sollte/hatte es auf WinCE zu laufen. Da aber kein Bedarf war – und andere Betriebssysteme/Gerätetypen sehr, sehr wichtig wurden (Android, Kobo) wurde m.E. dort nichts weiter verfolgt.
* Mittlerweile sind äußerst viele Komponenten hinzugekommen (Jasper, MapServer, …), aber die wurden nicht auf MSCV-Tauglichkeit getestet bzw. angepasst.
* Da ich diese Komponenten nicht kenne, fällt es mir im Nachgang sehr schwer, alles richtig anzupassen!

20.03.2020, 22:00 erste Erfolgsmeldung:

\* **am 20.03.2020 um 22:00 gab es dann auch ein lauffähiges Programm XCSoar, compiliert mit VisualStudio 2019 über CMake**

- kleine Bildabmessungen beim Start (800x600?)

- Aufgabe Pirna-Eisenach-Burg mit Linie dargestellt, WP waren aber nicht bezeichnet

- 4 Linien von (0,0) zu 4 Punkten in der Karte?

- kein Airspace dargestellt

- MouseScrolling sehr holprig

- MouseGesten scheinen zu funktionieren

- kein MouseHover (keine Einblendung der WP usw.)

- WP: UTF-8 als ANSI

- Menüsprache: English statt Deutsch

20.03.2020, 17:00Uhr

Project ist jetzt mit MSVC ohne Fehler compilierbar, aber

- erst war die main-Funktion vin lua noch aktive (lua.c und luac.c müssen raus!)

- Screen hat Größe 120x0 - das geht nicht (Workaround: y auch auf 120 gesetzt)

- Font ist nicht gültig (Workaround: Test wird übersprungen)

- Bitmap klappt noch nicht richtig (erfordert noch etwas Mehrarbeit, da habe ich einige - wahrscheinlich falsche - Flags eingebaut)

- die Resourcen werden nicht geladen\_

- überhaut erst einmal das RC-File (XCSoar.rc) ins Project aufgenommen

- die Macros gehen nicht richtig (bei BITMAP\_ICON wird der 'file'-Name nicht ins Macro geschoben!

21.03.2020, weitere Ziele

- Die Resourcen in XCSoar.rc wieder per Macro definieren (die Datei-Pfade hatte ich aufgelöst, weil das Concatening (Zusammenfügen) nicht richtig geklappt hat…

- PNG-Resourcen werden derzeit nicht geladen ;-(

**Vorgehensweise 3rd-Party Packages**

* Load der Sources (von GitHub,…) in D:\Projects\3rd\_Party
* Run CMake mit Binary-Directory D:\Projects\Binary\3rd\_Party (entspricht bei BO D:\build\_dir)
* (CMake-)Install der Export-Dateien in D:\Projects\link\_libs (Oder besser gleich in D:\link\_libs wie bei Baumer?

**Nächste Ziele**

* Prio 1) Resourcen- Bereinigen (Löst vielleicht das Airspace-Problem)
* Prio 1) Das Casting PixelSize / ??? ist noch falsch (Workaround MSC\_AUG)
* Prio 2) das 3rd-Party Konzept konsequent durchziehen (z.B. Qt, hier wird noch der Install-Step benötigt!)
* Prio 3) Übersetzen mit MinGW, eventuell in Eclipse
* Prio 4) Übersetzung mit CMake in Linux auf Linux
* Prio 4) Übersetzung mit CMake in Linux auf Windows (Crossbuild Toolchain)
* Prio 4) Übersetzung mit CMake in Android auf Windows (Crossbuild Toolchain)
* Prio 4) Übersetzung mit CMake in Kobo auf Windows (Crossbuild Toolchain)
* Prio 9) bestehendes Programm mit Qt ‚unterlegen‘
* Prio 10) neue Menü-Struktur überlegen und mit Qt abbilden

**Vorgehensweise 3rd-Party Packages – Update 11.04.2020**

* Vieles von meinen Vorgedanken ist heute obsolet: Durch das CMake-Package ‚**ExternalProject**‘ wird bereits (fast) alles abgefangen, was ich mir aufbauen wollte. Hier kann man 3rd-Party-Projekte als Zip-File herunterladen, entpacken – und dann damit arbeiten – man kann aber auch gleich mit Git sich verkoppeln – und durch ein Tag auf einen bestimmten Versionsstand aufsetzen… Damit kann man die externen Projekte intern einbinden (eben als ‚external Project‘) oder mit find\_Package die im System bereits installierten Packages koppeln, als Beispiel wird im Net auf **openchemistry** verwiesen, wo ein sogenanntes **superproject** beide Möglichkeiten vereint. Ganz verstanden habe ich zwar noch nicht, wie es dort gelöst ist, aber m.E. wird über die OPTION-Flags (USE\_SYSTEM\_XXX, z.b. USE\_SYSTEM\_ZLIB) gesteuert, ob ein 3rd-Party-Paket als ‚internes‘ ExternalProject‘ eingebunden wird – oder ob über es als ‚System-Bibliothek‘ über find\_package gesucht wird. Letztendlich könnte man über diese Flags auch steuern, dass in einem ersten Durchlauf die Pakete gebildet werden und bei allen weitern Läufen das find\_package herangezogen wird…
* Das will ich auf alle Fälle erst einmal so probieren!