Building WoG 3.59 Source

by Jakub Grzana

Prerequisites. What will you need:

- WoG source: ORIGINAL MIRROR
- Visual Studio Express Edition: OFFICIAL MIRROR
- Visual Studio Service Pack 1: OFFICIAL MIRROR
- Windows SDK v7.1: OFFICIAL MIRROR-RAW
- BuildLog for successful building: CLICK

Visual Express Ed and SP1 installation:

Installation of Visual Express isn't rocket science, if you face any problems (which i didn't) try running with admin permissions and in compatibility mode. Remember path to which you install your visual express, from now on I will refer to as **VISUAL_PATH**.

To install SP1 (if you use official link) download all .exe files available and try which one will work.

This is not the end though, you need to go into **VISUAL_PATH**.. You should see following folders:


Visual is supposed to be used with admin permissions, so you need to find and change properties of exec. You can find it in **VISUAL_PATH**.`/Common7/IDE/VCExpress.exe`

In catalog Microsoft Visual C++ 2005 Express Edition you may find `setup.exe` and all files required to repair or reinstall your Visual. Use it if you screw something up.

To run Visual, search for VCExpress.exe or just start program in `IDE` catalog.
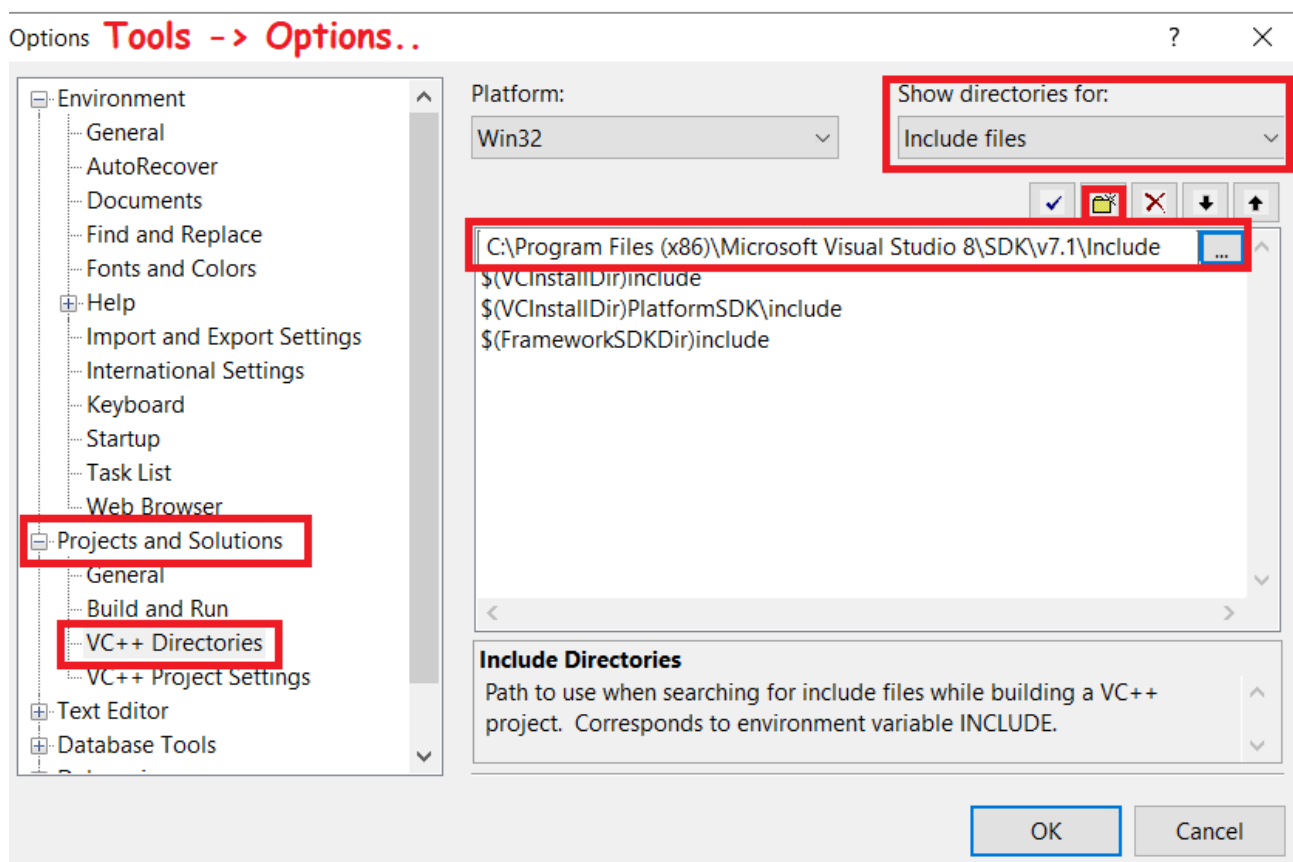
Installation of SDK:

Using official link is preferred. Installation is simple, at the beginning you may be warned:
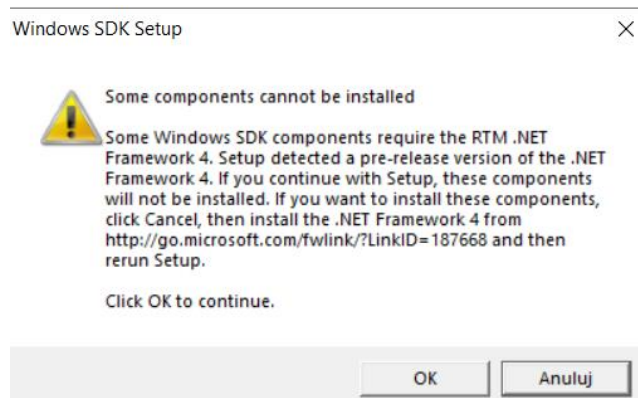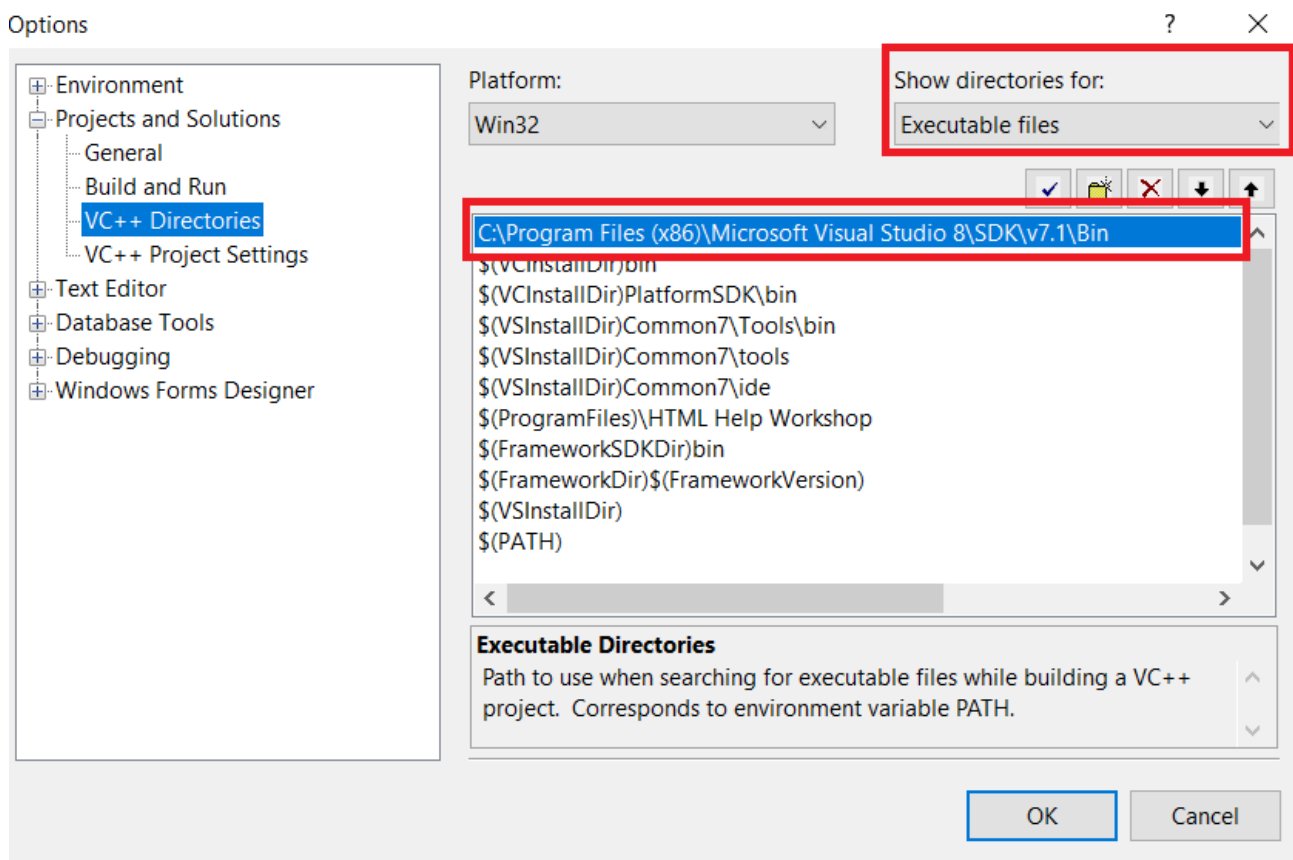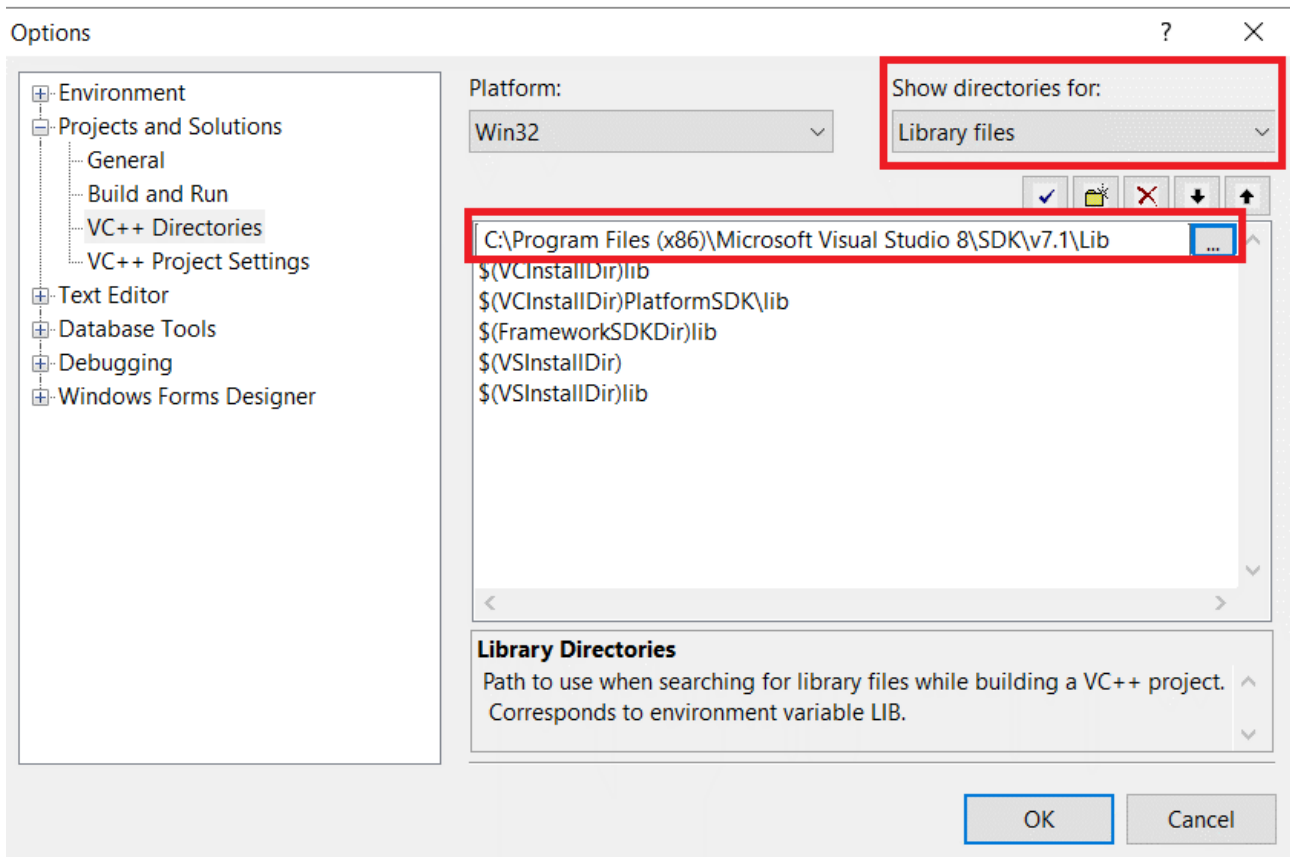

That's okay, just ignore and go continue. Make sure to install Inc(lude), Lib(rary) and Bin/Executable files for architecture x86. If you don't know what that means, just don't change default setup. Remember installation path, I will refer to that as **SDK_PATH**.

However if you use MIRROR-RAW link then there is no installation, those are just raw files. Unpack archive and move v7.1 folder to **VISUAL_PATH**/SDK.

Regardless of used way, this isn't everything. You still need to „link" Visual to your SDK installation. Run VCExpress.exe and on topbar select Tools→Options. Then add paths to your SDK to include files, library files and executable files. All you can found in **SDK_PATH/v7.1**

Options

Platform:
Win32

Show directories for:
Library files

C:\Program Files (x86)\Microsoft Visual Studio 8\SDK\v7.1\Lib   ...
$(VCInstallDir)lib
$(VCInstallDir)PlatformSDK\lib
$(FrameworkSDKDir)lib
$(VSInstallDir)
$(VSInstallDir)lib

- Environment
- Projects and Solutions
  - General
  - Build and Run
  - VC++ Directories
  - VC++ Project Settings
- Text Editor
- Database Tools
- Debugging
- Windows Forms Designer

Library Directories
Path to use when searching for library files while building a VC++ project. Corresponds to environment variable LIB.

OK   Cancel

Options

Platform:
Win32

Show directories for:
Executable files

C:\Program Files (x86)\Microsoft Visual Studio 8\SDK\v7.1\Bin
$(VCInstallDir)bin
$(VCInstallDir)PlatformSDK\bin
$(VSInstallDir)Common7\Tools\bin
$(VSInstallDir)Common7\tools
$(VSInstallDir)Common7\ide
$(ProgramFiles)\HTML Help Workshop
$(FrameworkSDKDir)bin
$(FrameworkDir)$(FrameworkVersion)
$(VSInstallDir)
$(PATH)

- Environment
- Projects and Solutions
  - General
  - Build and Run
  - VC++ Directories
  - VC++ Project Settings
- Text Editor
- Database Tools
- Debugging
- Windows Forms Designer

Executable Directories
Path to use when searching for executable files while building a VC++ project. Corresponds to environment variable PATH.

OK   Cancel

Environmental variable

This isn't required to successfully build WoG source, but very handy in usage. Cool step-by-step tutorial can be found here, I will just post some pictures taken from that.
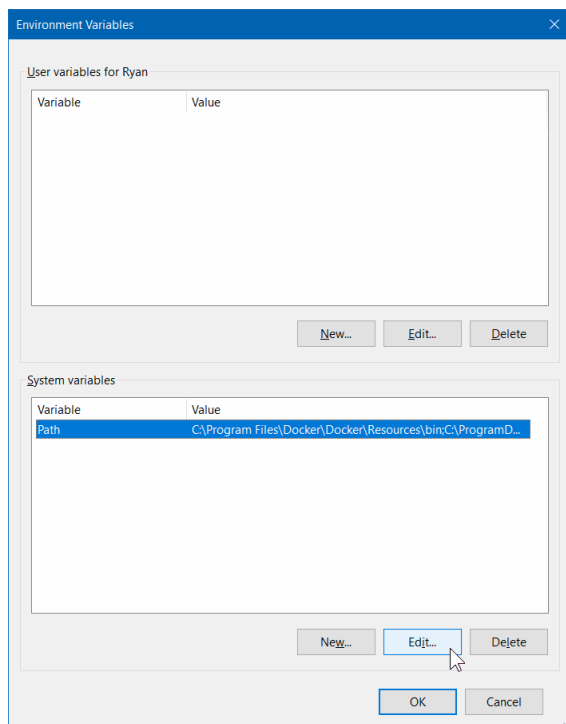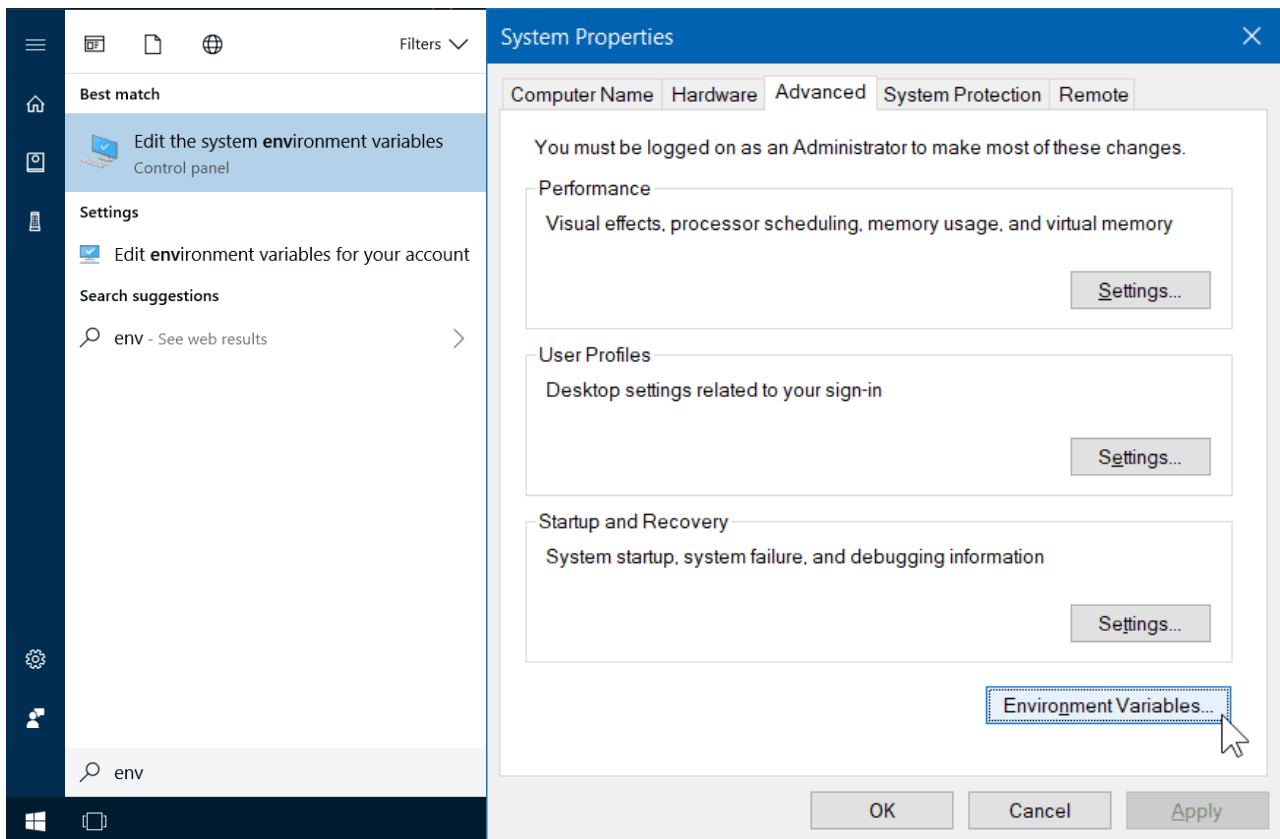
Create new env-var, named `HOMMWOGDIR`
As value paste path to directory where you want to place compiled & built WoGT1.exe – typically HoMM directory for playtesting.
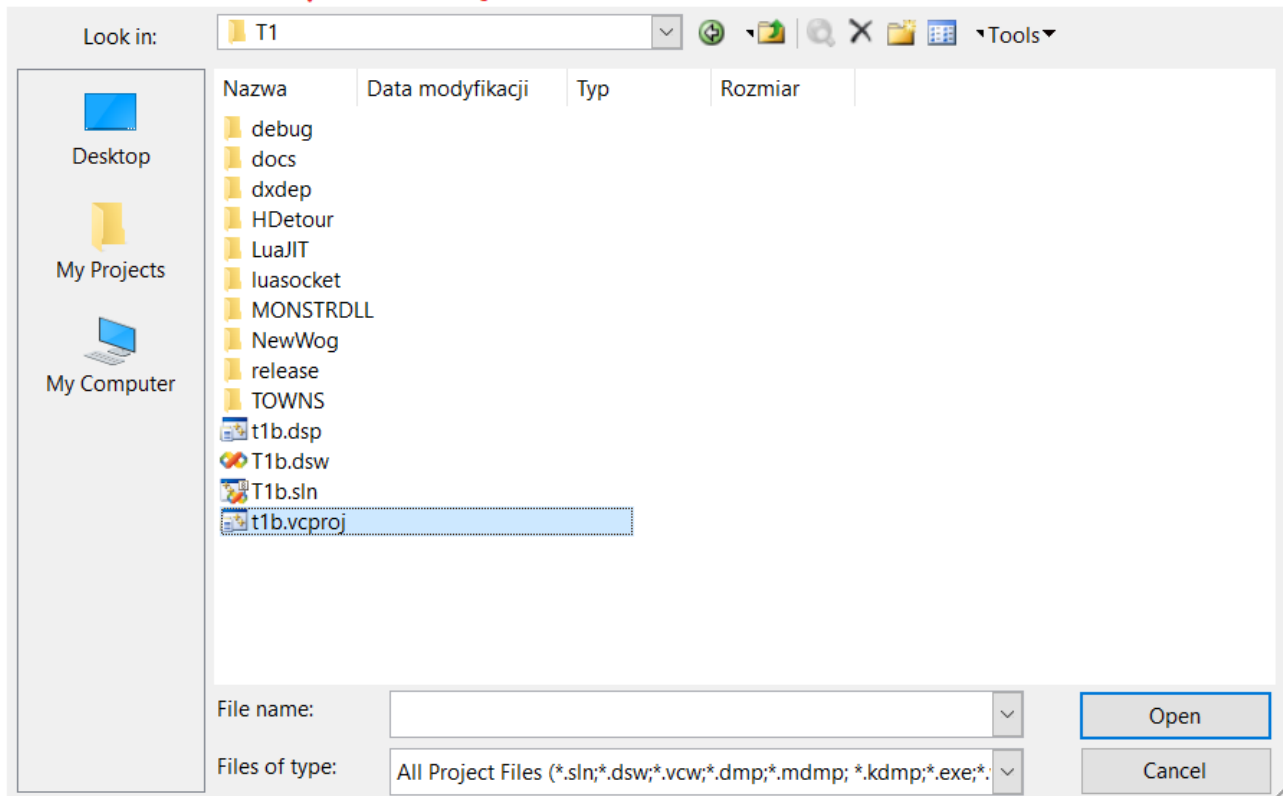
After adding or editing env-variable **restart Visual, if it is running**.

Load WoG project

To load project, run VCExpress.exe and open solution t1b.vcproj

Original source released by GrayFace has some bugs. Quick fixes:

In solution explorer, open t1b, header files, common.h. Find `//#define STOP`. This is macro we need, but it is commented. Thus, uncomment that.
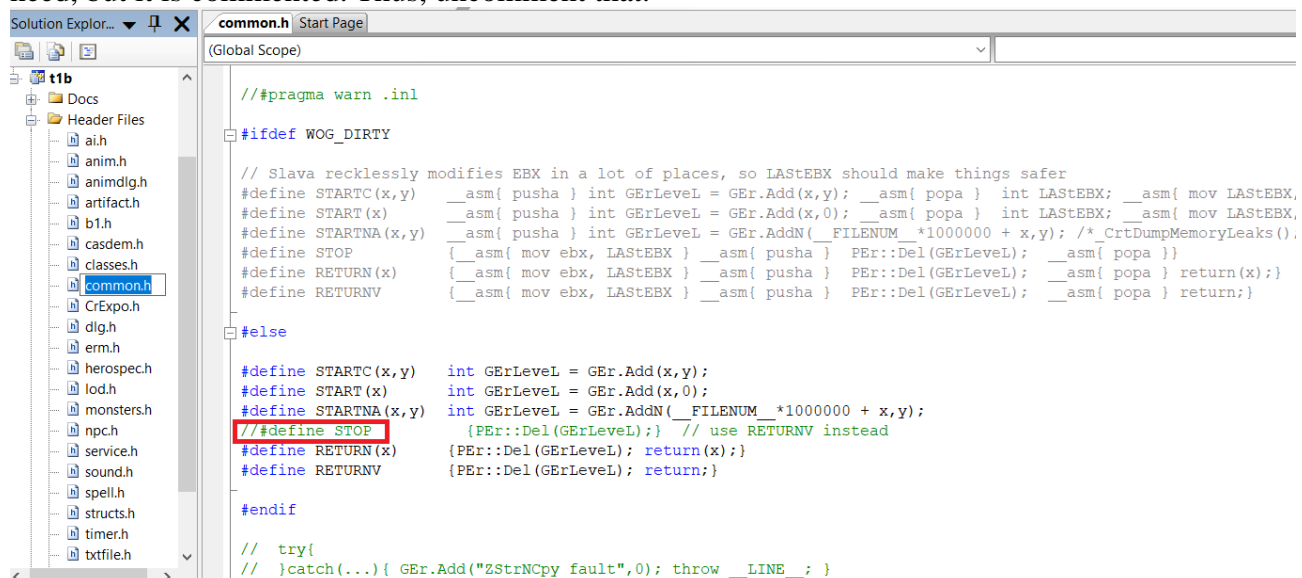
In solution explorer, open t1b, source files, _B1.cpp. Paste these two lines just after #include block

```
// TEMPORARY
#pragma comment(lib, "User32.lib")
#pragma comment(lib, "Gdi32.lib")
```
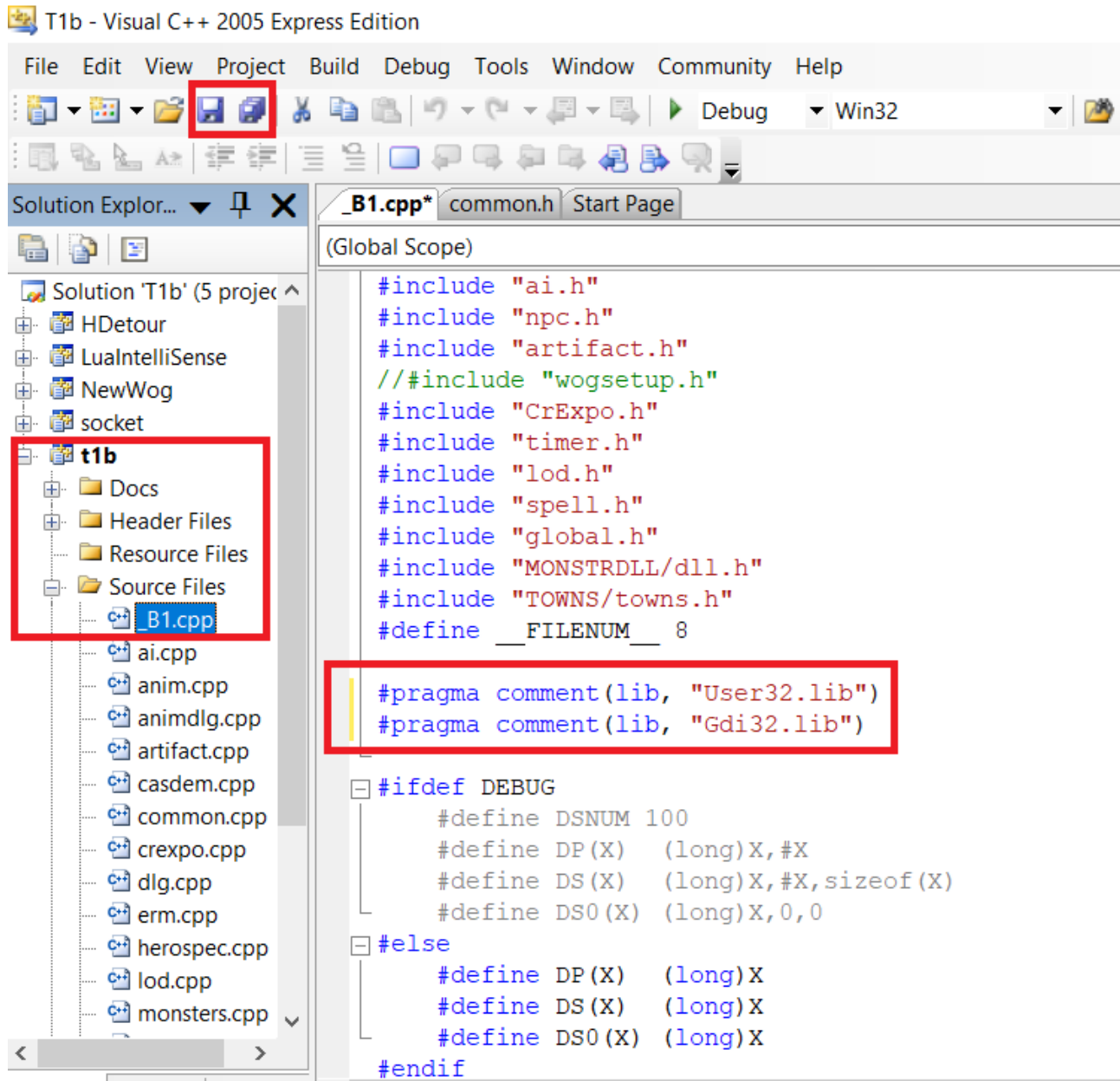

This prevents nasty linking errors (unresolved external symbol) where linker doesn't link User32.lib and Gdi32.lib despite properly linked SDK directories. I don't think this is proper solution, but works for now.

Configurations

Release configuration is used for deployment, Dll for debugging. For now, change all to Release.


```
#include "timer.h"
#include "lod.h"
#include "spell.h"
#include "global.h"
#include "MONSTRDLL/dll.h"
#include "TOWNS/towns.h"
#define __FILENUM__ 8
```
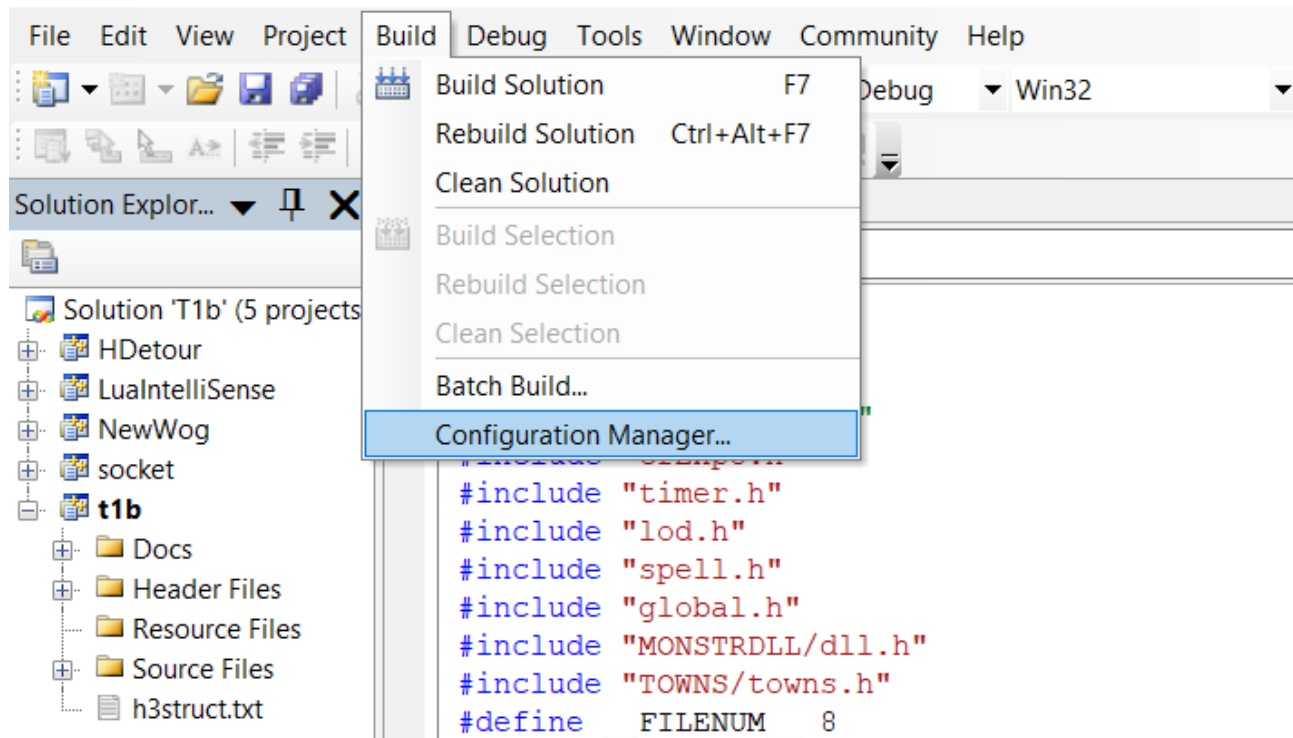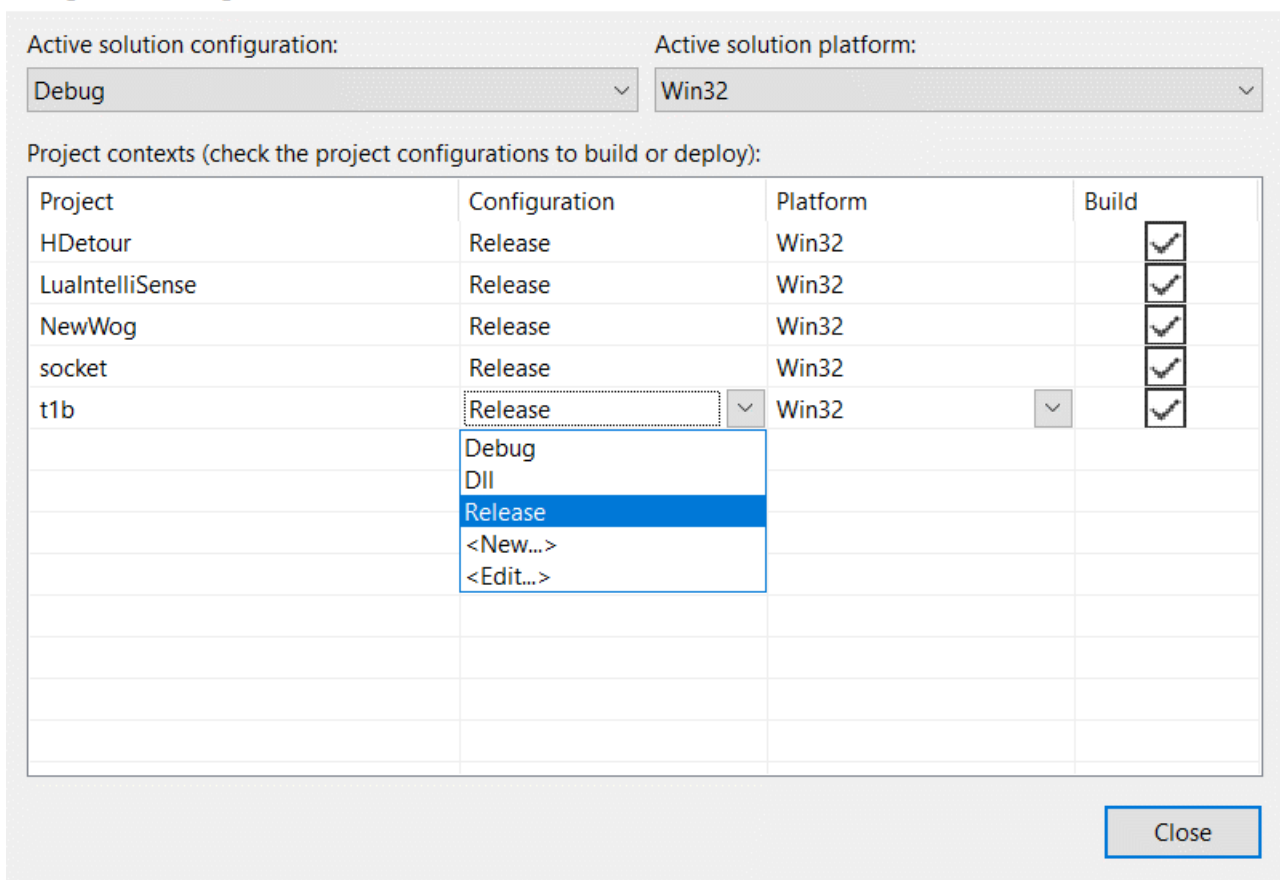
Building

Load project, rightclick Solution T1b in solution explorer, then build solution. It may take few minutes
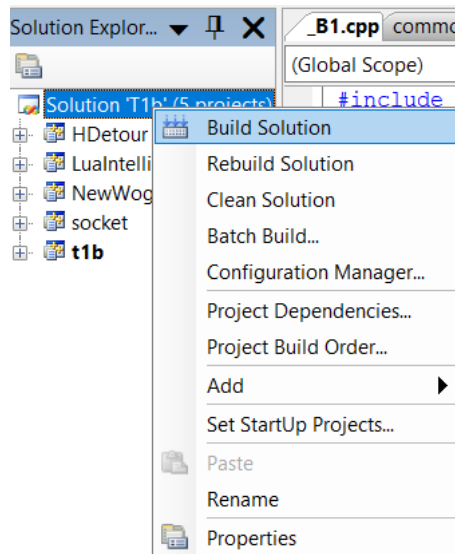
Project will automatically compile, build and inject code into Heroes 3 SoD exec (it is stored in wog-master/Build/ with changed name, don't worry about that) New exec will be named WogT1.exe and placed in path stored in HOMMWOGDIR variable.

If there is problem with env-var, exec will (probably) be placed in C:/ dir.

Buildlog for successful building process can be found in Prerequisites section, for you to compare. Download and open in browser. There are many warning, but that's normal.

Creating release

Exec is not everything, there are additional files required. All of them are placed in wog-master, although not in proper paths. [There](#) are some examples of properly prepared releases.

Wog-master folder   Release

Copy: Data, Mods, WogDialogs.dll, FASM.dll, BINKW32new.dll to main directory of new release. Move here WogT1.exe aswell. Go inside Data/ folder and create folders: Data/zvs/Lua. Copy all scripts from wog-master/Lua to Data/zvs/Lua. Pack everything to archive for easy installation. That's it.

Installation: Extract contents of release-archive to WoG home directory. Using a folder with clean WoG 3.58 install is required.

Some other informations about WoG 3.59 and project

Exec is created using C/C++ code that is compiled and injected into H3 SoD exec kept inside wog-master/Build. To change game's behaviour creators used hooks and many more low-level hacking tricks.

WoG is written (generally) in 3 languages:
- C/C++ used to make .exe file. Check wog-master/T1/ to understand more.
- Lua, internal scripts, used as add-on to exec. Allows basically everything that C/C++ does, but without need to "build". Although things that are already changed (like ERM parser) shouldn't be changed that way. Check wog-master/Lua/ for more
- Lua+ERM, external scripts, used to add new features, like new creature banks, modifying secondary skills and so on. Check /Mods/WoG to understand more

ERM Help coverin both ERM and basics of Lua, can be found [here](#).