Lesson07

Table des matières

Introduction

Bonjour,

J'ai beaucoup appris en essayant les exemples de sources OpenGL de NeHe (Neon Helium).

Accueil

Ces sources, même s'ils commencent à dater un peu, sont une mine pour comprendre les concepts de base d'OpenGL sur environnement Windows.
Alors je vous souhaite de profiter autant que moi de cet apprentissage, et je me suis imposé pas mal de contraintes.
En voici, quelques unes :
- faire converger les sources des applications pour n'obtenir qu'une seule version de ces sources compatibles avec TOUS les environnements de développements et les compilateurs C/C++ "free" disponibles sur Windows
- utiliser au début le même IDE (.ie. Code::Blocks) pour générer TOUS les exécutables de ces applications en version 32 et 64 bits, aussi bien en version Debug que Release.

Ce qui a conduit, en utilisant un "virtual target" de cet IDE dénommé "All build", a pouvoir générer "automatiquement" 52 exécutables correspondant à 26 environnements de développement différents.

Voici la liste de tous ces environnements (même si la génération Open Watcom 64 bits pose problème à l'IDE CB !) :
Borland C/C++ 5.51, CLANG adossé à Mingw32 (issu de Winlibs), CLANG adossé à Mingw64 (issu de Winlibs), CLANG adossé à MSYS2/Mingw32, CLANG adossé à MSYS2/Mingw64, CLANG adossé à Visual Studio 2022 32 bits, CLANG adossé à Visual Studio 2022 64 bits, un IDE Red Panda DevCpp installé avec une version de Mingw64, l'IDE Code::Blocks installé avec une version de Mingw64, Mingw32 version officielle, GCC inclus dans MSYS2/Mingw32, GCC inclus dans MSYS2/Mingw64, GCC inclus dans CYGWIN64/Mingw32, GCC inclus dans CYGWIN64/Mingw64, GCC inclus dans TDM32/Mingw32, GCC inclus dans TDM64/Mingw64, Mingw32 issu du package WinLibs, Mingw64 issu du package WinLibs, LCC version 32 bits, LCC version 64 bits, Pelles C version 32 bits, Pelles C version 64 bits, Open Watcom version 32 bits, Open Watcom version 64 bits, Visual Studio 2022 "community" en mode 32 bits, Visual Studio 2022 "community" en mode 64 bits.

Quoi de neuf

Mais comme j'ai aussi souhaité me frotter à un autre utilitaire de génération automatique dénommé CMAKE (pour mémoire CMAKE génère automatiquement des fichiers Makefile qu'il faut donc ensuite lancer en mode classique), j'ai aussi créé l'ensemble des fichiers de paramétrages utiles, dans le répertoire build.cmake.
Et bien entendu, il reste quelques difficultés/bugs avec CMAKE, que j'ai résolu par l'utilisation de fichiers Makefile en mode natif pour chaque environnement de développement non traité correctement par CMAKE (.ie. Borland C/C++, Digital Mars C/C++, LCC32, LCC64, Pelles C 32 bits, Pelles C 64 bits, Open Watcom C/C++ 32 bits, Open Watcom C/C++ 64 bits) en présence de fichier Ressource : (*.rc).

Comment démarrer une génération ?

Exigences du système

Bien sur, il faut avoir au préalable installer de façon "propre" TOUS ces différents de développement.

On commence, par Code::Blocks (origine SourceForge ou le site Web CB disponible), Red Panda DevCpp (origine SourceForge, LLVM en version 32 et 64 bits (pour CLANG adossé à Visual Studio), Borland C/C++, les deux package WinLibs de très bonne qualité, Mingw32 en version officielle (origine SourceForge), MSYS2 en procédant ensuite à l'installation des environnements de développement Mingw32 et Mingw64, mais aussi CLANG 32 bits et CLANG 64 bits, Mingw64 (origine SourceForge), le package CYGWIN64 en procédant ensuite à l'installation des environnements de développement Mingw32 et Mingw64 (sans oublier l'utilitaire "make" pour CMAKE !), LCC 32 bits, LCC 64 bits, Pelles C (on installe les deux versions 32 et 64 bits) et Open Watcom C/C++ (on installe les deux versions 32 et 64 bits), TDM 32 bits, TDM 64 bits, Visual Studio 2022 community (on installe les deux versions 32 et 64 bits) sans oublier un Kit de développement Windows compatible avec votre environnement et votre version "système".

Et, comme si cela ne suffisait pas, il faut aussi mettre à niveau les "include" et parfois les "lib" d'OpenGL pour certains compilateurs (et pas des moindre, celui de M$ par exemple : VS2022, sans oublier le plus ancien DMC ...).

Vous trouverez toutes ces évolutions supplémentaires dans mon projet "tools_tde", toujours sur GITHUB, dans le sous répertoires "Modifs_compilers".

Ce qui donne le descriptif suivant :

OpenGL Tutorial #6.

Project Name: Jeff Molofee's OpenGL Tutorial

Project Description: Texture Mapping Tutorial

Authors Name: Jeff Molofee (aka NeHe)

Authors Web Site: nehe.gamedev.net

COPYRIGHT AND DISCLAIMER: (c)2000 Jeff Molofee

> If you plan to put this program on your web page or a cdrom of
> any sort, let me know via email, I'm curious to see where
> it ends up :)
>
> If you use the code for your own projects please give me credit,
> or mention my web site somewhere in your program or it's docs.

Modified smoothly by Thierry DECHAIZE

Paradigm : obtain one source (only one !) compatible for multiple free C Compilers
and provide for all users an development environment on Windows 11 64 bits
the great Code::Blocks manager (version 20.3), and don't use glaux.lib or glaux.dll.

a) Mingw 32 bits, version officielle gcc 9.2.0 (old !) : downloadable on http://sourceforge.net/projects/mingw/ (official)
b) Mingw 64 bits included in new IDE Red Panda Dev-Cpp, version gcc 10.3.0 : donwloadable on http://sourceforge.net/projects/dev-cpp-2020/
c) Mingw 64 bits included in package Code::Blocks (version 20.03), version gcc 8.1.0 (very old !) : downloadable on http://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/
d) Mingw 32 and 64 bits packagés, version gcc 11.2.0 : downloadable on https://winlibs.com/ (and CLANG included in, 32 and 64 bits), two kits :
- winlibs-i686-posix-dwarf-gcc-12.2.0-llvm-14.0.6-mingw-w64ucrt-10.0.0-r2.7z (32 bits)

- winlibs-x86_64-posix-seh-gcc-12.2.0-llvm-14.0.6-mingw-w64ucrt-10.0.0-r2.7z (64 bits)

e) Cygwin64, 32 et 64 bits, version gcc 11.3.0 : downloadable on http://www.cygwin.com/install.html (tool for install : setup-x86_64.exe)

f) TDM GCC, 32 et 64 bits, version 10.3.0 : downloadable on http://sourceforge.net/projects/TDM-GCC

g) MSYS2 environnement MINGW32 and MINGW64, 32 et 64 bits, version de 2022 (msys2-x86_64-20220603.exe), version gcc 12.2.0 : downloadable on https://repo.msys2.org/distrib/x86_64/msys2-x86_64-20220603.exe

h) Visual Studio 2022, 32 et 64 bits, community edition for free : downloadable on https://visualstudio.microsoft.com/fr/thank-you-downloading-visual-studio/?sku=Community&rel=17

i) Borland C/C++ 32 bits, version 5.5 : downloadable on https://developerinsider.co/download-and-install-borland-c-compiler-on-windows-10/

j) Digital Mars Compiler C/C++ 32 bits version 8.57 : downloadable on http://www.digitalmars.com (the more old compiler, the more bugged, dead branch !)

k) OpenWatcom C/C++ 32 et 64 bits, version 2.0 : downloadable on http://openwatcom.mirror.fr/ (only 32 bits version run correctly !)

l) Lcc and Lcc64, 32 et 64 bits: downloadable http://www.cs.virginia.edu/~lcc-win32/

m) PELLES C (only C) , 32 et 64 bits, version 11.0 : downloadable on http://www.smorgasbordet.com/pellesc/

o) CLANG, adossé aux environnements MINGW64 et MINGW32, version 14.0.6 (version gcc 12.0.0) : downloadable on https://winlibs.com/

p) CLANG, adossé aux environnements Visual Studio 2022 (+ kits Microsoft), version 15.0.0 : downloadable on https://releases.llvm.org/download.html

q) CLANG de la version MSYS2, adossé aux environnements MINGW64 et MINGW32, version 15.0.0 (version gcc 12.2.0) : downloadable on https://repo.msys2.org/distrib/x86_64/msys2-x86_64-20220118.exe

r) CLANG de la version CYGWIN, adossé aux environnements MINGW64 et MINGW32, version 8.0.0 (very old !) (version gcc 11.3.0) : downloadable http://www.cygwin.com/install.html (tool for install or update : setup-x86_64.exe)

TDE -> Add resource file and resource header for restitute version + icon OpenGL.ico for fun because versionning is important, same for freeware :-) !

Lancement d'une génération CB ou CMAKE

Pour générer les exécutables attendus, plusieurs options :

- la première très simple, on ouvre le wokrspace ou le projet CB (.ie. "Lesson"nn".workspace" or "Lesson"nn".cbp"), puis on clique sur "rebuild", en ayant sélectionné soit la "virtual traget" -> 'All build', soit un des "real target" : 'Debug DMC' (par exemple),
- la deuxième est d'utiliser le fichier de commande fourni : generate_all_with_cmake.bat avec comme paramètres :

a) le premier paramètre : le répertoire "master" ou se trouve votre projet (.ie. C:\src\OpenGL\NeHe_Lesson01-master par exemple, attention, ce n'est pas le répertoire des sources qui se trouvent dans un sous-répertoire "src" de ce "master")

b) le deuxième paramètre : le nom de votre application (.ie. "Lesson01" par exemple)

c) le troisième paramètre (optionnel) permet de choisir alors la génération souhaitée parmi une liste d'environnements de compilation/linkage des exécutables.

Il suffit de lancer ce fichier de commande sans paramètre pour obtenir l'explication des différents choix de paramètres.