

Tutorial – Microsoft Windows

Crearea unui proiect Microsoft Visual Studio pentru o aplicație OpenGL/GLFW

Acest tutorial are ca scop familiarizarea cu mediul de dezvoltare Microsoft Visual Studio și cu pașii necesari creării unui proiect nou pentru o aplicație OpenGL care utilizează biblioteca GLFW pentru gestionarea ferestrei și a evenimentelor generate de dispozitivele de intrare (mouse, tastatură).

1. Biblioteca GLFW

Pentru a putea dezvolta o aplicație folosind biblioteca GLFW este nevoie să se obțină fișierele header (extensia .h) ale bibliotecii și fișierele binare (extensia .lib) ale acesteia.

Metoda recomandată pentru a obține fișierele binare este de a compila întreaga librărie din surse. Doar compilarea din surse asigură compatibilitate maximă între librărie și platforma pe care se vor dezvolta aplicațiile. Alternativ, binarele precompilate pentru diferite platforme predefinite pot fi descărcate de pe website-ul <https://www.glfw.org/>.

1.1. Compilarea bibliotecii din surse

- Sursele bibliotecii GLFW pot fi descărcate de pe website-ul <https://www.glfw.org/download.html> (Figura 1)

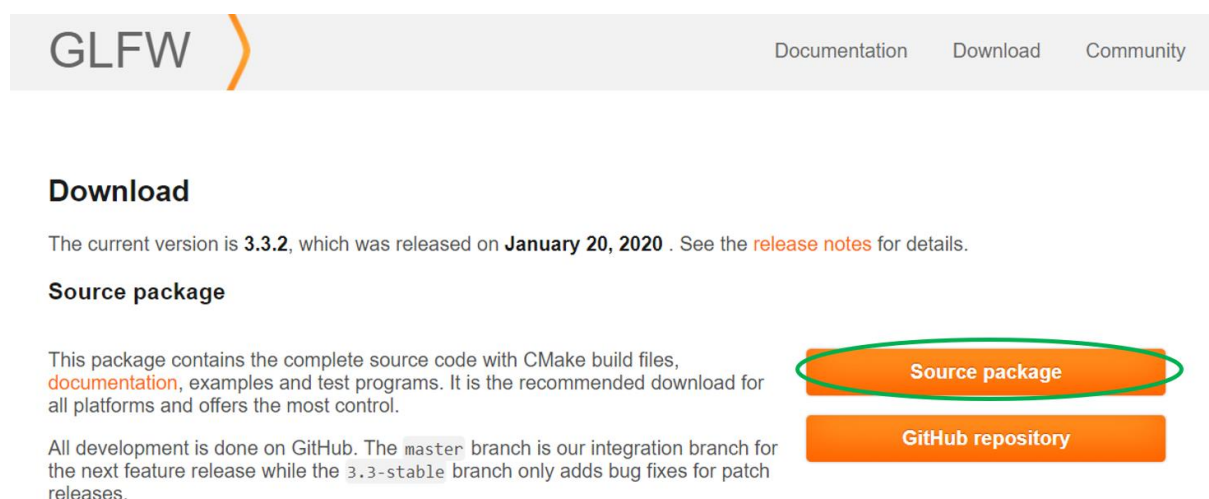


Figura 1 - Descărcarea surselor bibliotecii GLFW

- Extrageți întreg conținutul arhivei folosind Windows Explorer sau aplicația voastră preferată pentru gestionarea arhivelor (ex. 7-zip, WinRar, etc).

Name	Date modified	Type	Size
glfw-3.3.2	20.01.2020 02:12	File folder	
glfw-3.3.2.zip	16.09.2020 09:39	zip Archive	1.401 KB

Figura 2 - Extragerea conținutului arhivei

- Pentru a putea compila sursele bibliotecii GLFW folosind Microsoft Visual Studio (MVS), descărcați și instalați CMake pentru Microsoft Windows <https://cmake.org/download/>. Pentru instrucțiuni despre instalarea Microsoft Visual Studio, mergeți la secțiunea 3.1

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.18.2.tar.gz
Windows Source (has \r\n line feeds)	cmake-3.18.2.zip

Binary distributions:

Platform	Files
Windows win64-x64 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.18.2-win64-x64.msi
Windows win64-x64 ZIP	cmake-3.18.2-win64-x64.zip
Windows win32-x86 Installer: Installer tool has changed. Uninstall CMake 3.4 or lower first!	cmake-3.18.2-win32-x86.msi
Windows win32-x86 ZIP	cmake-3.18.2-win32-x86.zip
Mac OS X 10.7 or later	cmake-3.18.2-Darwin-x86_64.dmg
	cmake-3.18.2-Darwin-x86_64.tar.gz
Linux x86_64	cmake-3.18.2-Linux-x86_64.sh
	cmake-3.18.2-Linux-x86_64.tar.gz

Figura 3 - Descărcarea și instalarea CMake

- Rulați aplicația CMake și selectați la rubrica „Where is the source code” calea spre folderul extras din arhivă (Figura 4). Selectați la rubrica „Where to build the binaries” calea către folderul *build* din folderul extras din arhivă (Figura 4). Chiar folderul *build* nu există, **scriți calea ca în Figura 4**, el va fi creat automat în timpul generării proiectului.

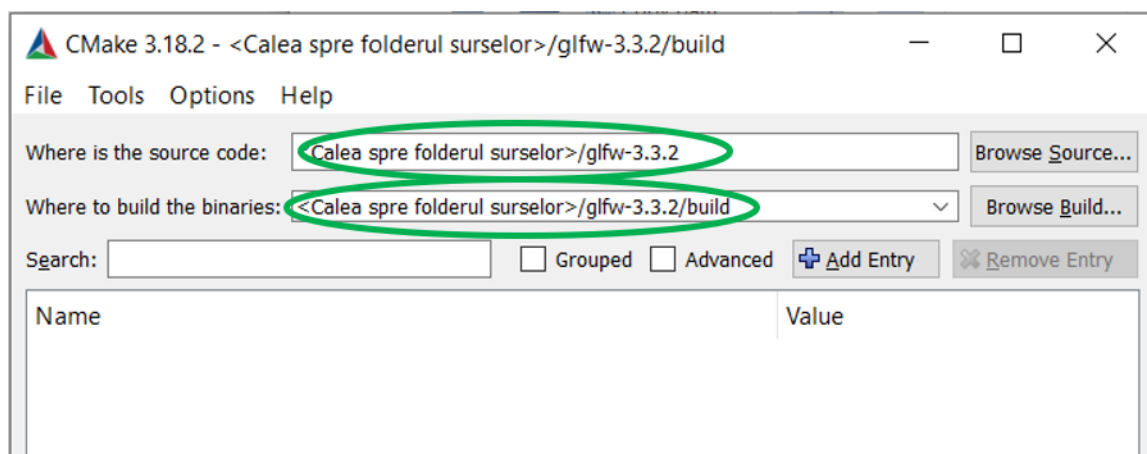


Figura 4 - Selectarea folderului sursă și a folderului în care se va genera proiectul MVS

- Apăsați Configure pentru actualizarea opțiunilor existente în generarea proiectului MVS (Figura 5)

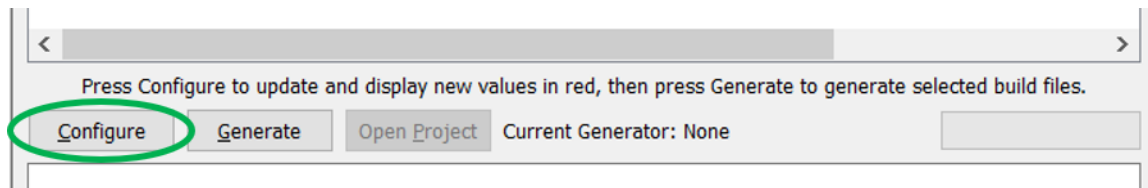


Figura 5 - Actualizarea opțiunilor pentru generarea proiectului MVS

- Selectați varianta de Microsoft Visual Studio pe care o aveți instalată (Figura 6). Dacă nu aveți instalat încă MVS, urmați pașii descriși în secțiunea 3.1 pentru instalare.

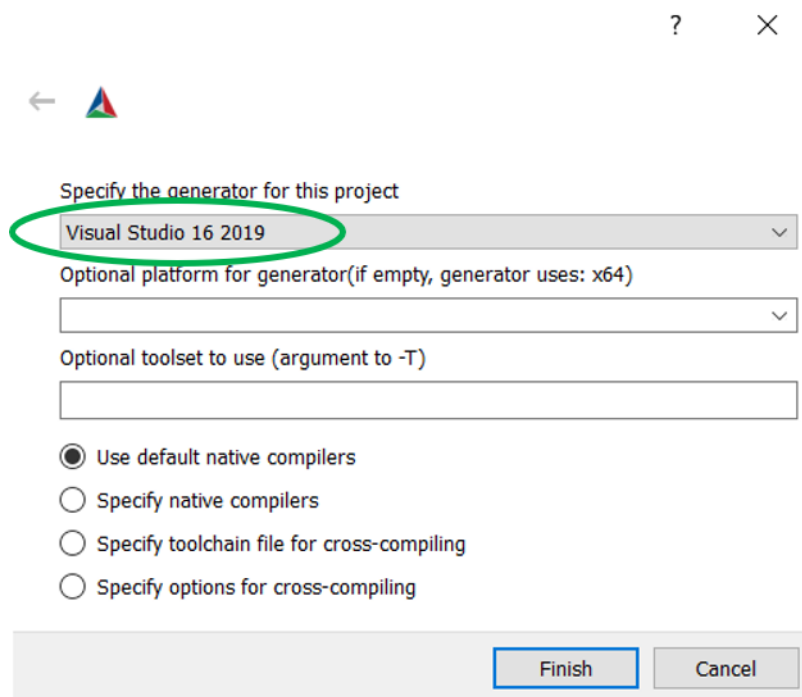


Figura 6 - Selectarea variantei de MVS instalate

- **Pentru utilizatorii avansați:** Dacă doriți, modificați starea implicită a opțiunilor de generare a proiectului MVS (ex. Dacă doriți generarea unor binare partajate (.dll) a bibliotecii GLFW, bifați opțiunea BUILD_SHARED_LIBS). **Dacă nu sunteți un utilizator avansat sau doriți compilarea bibliotecii GLFW doar pentru a fi utilizată pentru proiectele necesare la laborator, lăsați opțiunile în starea lor implicită!**
- Apăsați **din nou** butonul Configure pentru a confirma opțiunile cu care se va genera proiectul MVS. Fundalul opțiunilor ar trebui să se schimbe din roșu în alb (Figura 7).

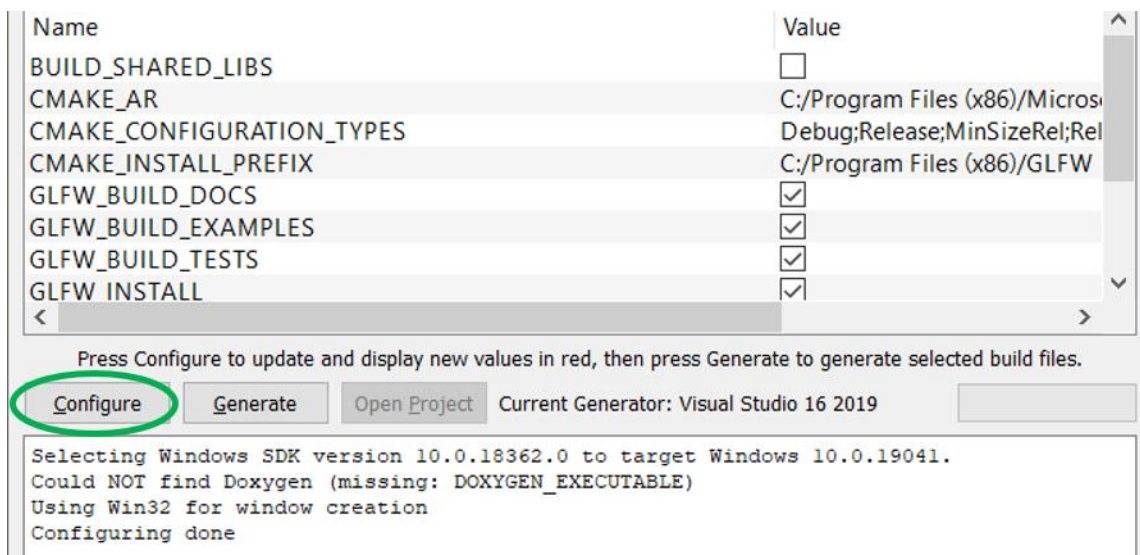


Figura 7 - Confirmarea opțiunilor pentru generarea proiectului MVS

- Dacă nu apar mesaje de eroare de configurare, în acest moment se poate genera soluția MVS care va fi utilizată pentru compilarea bibliotecii GLFW apăsând butonul Generate (Figura 8)

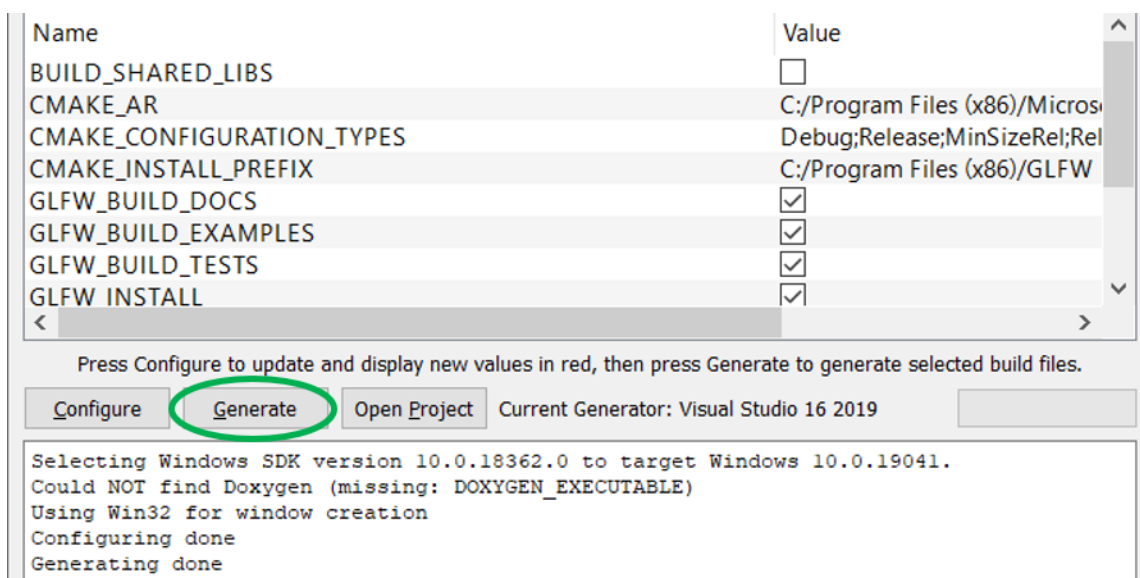


Figura 8 - Generarea proiectului MVS

- Dacă nu există erori, CMake va genera soluția și proiectul MVS cu numele GLFW.sln în folderul *build* din folderul care conține sursele bibliotecii. Dacă MVS este instalat, proiectul poate fi deschis direct din aplicația CMake, folosind butonul Open Project (Figura 9).

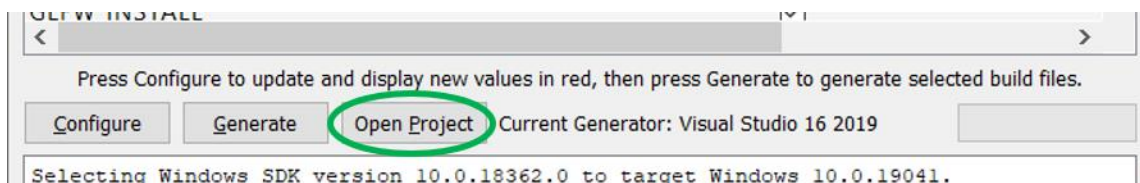


Figura 9 - Deschiderea proiectului MVS

- Selectați configurația dorită (ex. Debug sau Release), selectați platforma dorită (ex. x86 sau x64) (se recomandă utilizarea platformei x64) (Figura 10) și selectați din meniul ferestrei MVS **Build -> Rebuild Solution**; **Important!** Configurația și platforma selectată acum trebuie să coincidă cu configurația și platforma cu care va fi creat proiectul OpenGL ulterior, altfel vor apărea mesaje de eroare! Întrucât s-ar putea ca proiectul OpenGL final să fie compilat atât în configurația Debug cât și în configurația Release, se recomandă să se genereze fișierele binare ale bibliotecii GLFW pentru ambele (ceea ce presupune două compilări diferite pentru proiectul curent: o dată în configurația Debug și platforma x64 și a doua oară în configurația Release și platforma x64).

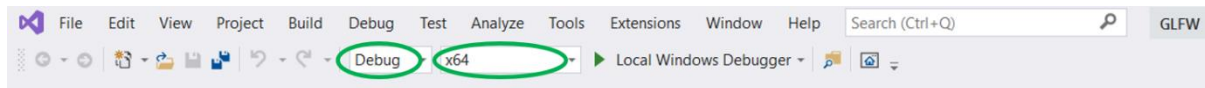


Figura 10 - Selectarea configurației și a platformei

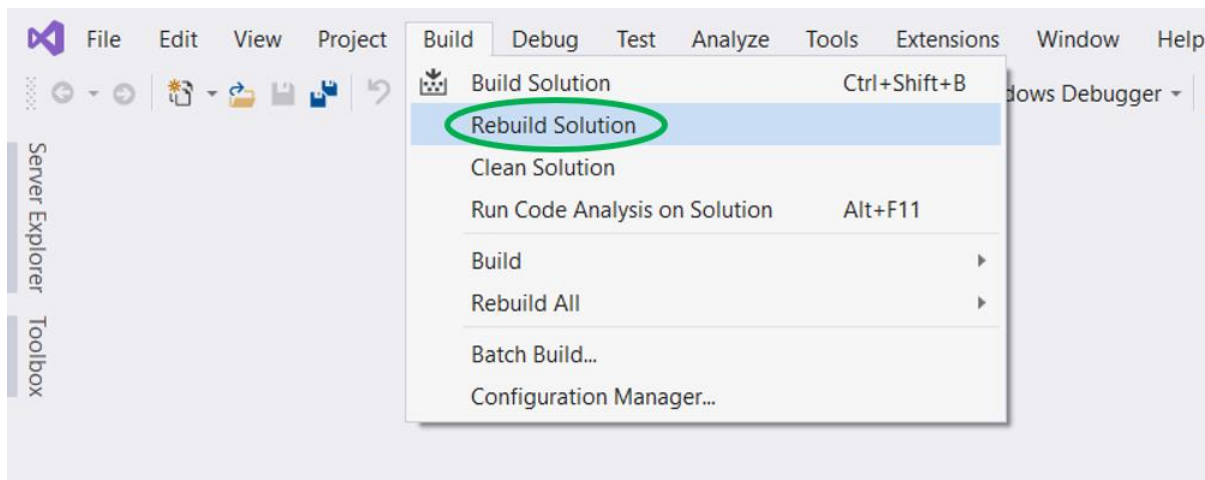


Figura 11 - Compilarea proiectului și generarea binarelor bibliotecii GLFW

- Fișierele header ale bibliotecii GLFW se vor găsi în folderul *include* din folderul rădăcină al bibliotecii (Figura 12), iar fișierele binare se vor găsi în folderul build->src-> <<numele configurației (Debug sau Release)>> (Figura 13).

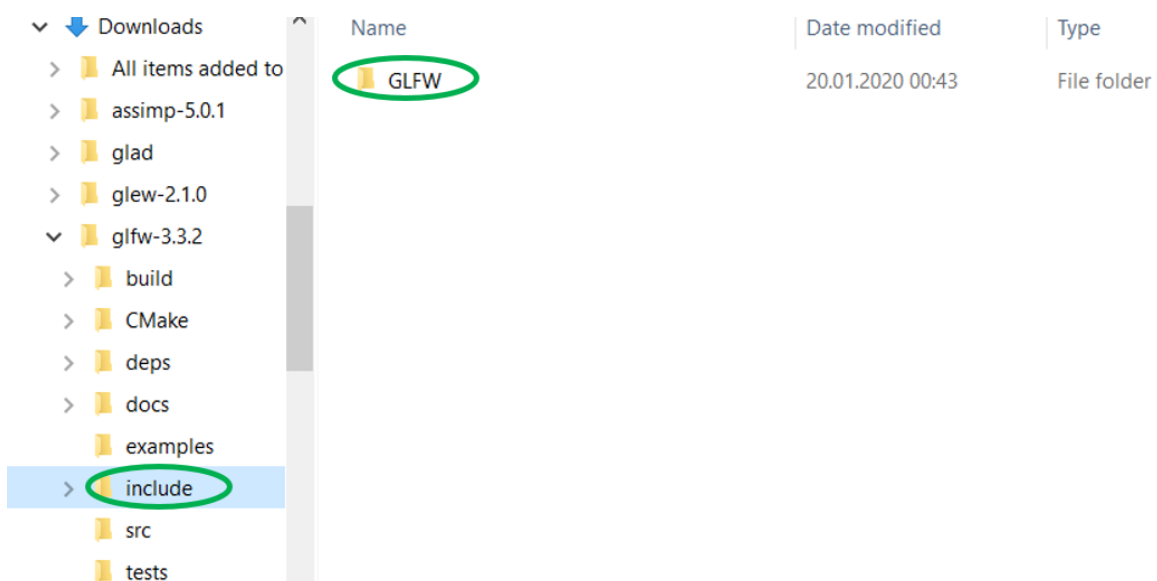


Figura 12 - Calea spre fișierele header

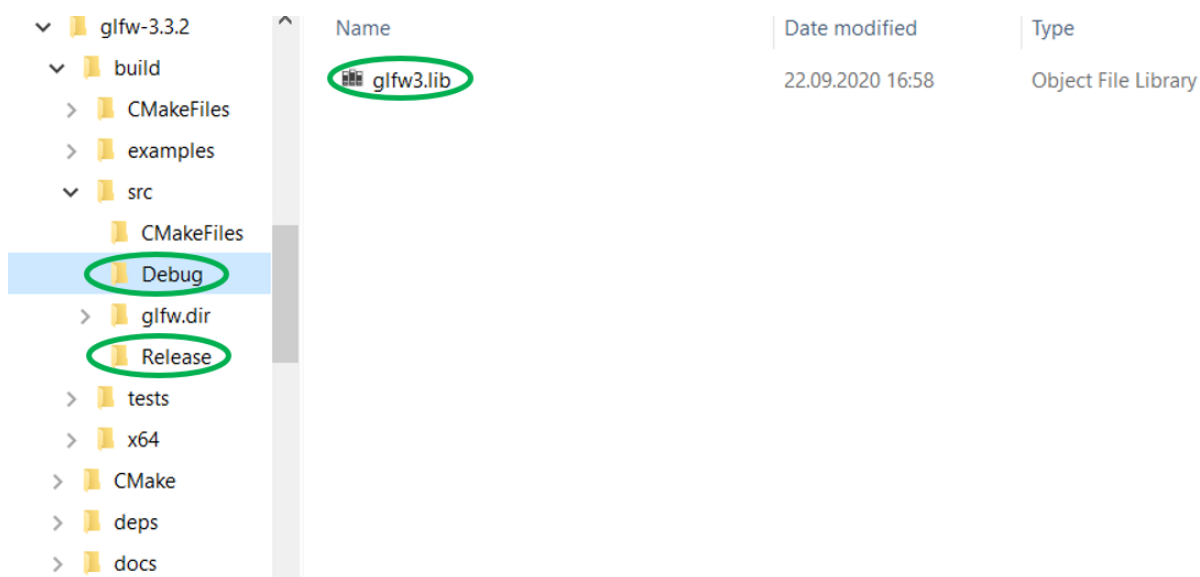


Figura 13 - Calea spre fișierele binare

1.2. Organizarea fișierelor librărilor auxiliare pentru aplicațiile OpenGL

- Fișierele header și cele binare pot fi copiate într-un loc din care să poată fi adăugate cu ușurință în toate proiectele OpenGL viitoare (ex. Într-un folder cu numele „OpenGL dev libs”). Se recomandă următoarea structură a folderului cu librării auxiliare: folderul va conține un subfolder cu numele „include” în care se vor pune toate fișierele header, fiecare librărie într-un folder separat (ex. fișierele header (extensia .h) ale bibliotecii GLFW vor fi puse într-un folder cu numele „GLFW” în interiorul folderului *include*) și un folder „lib” cu două subfoldere, „Debug” și „Release” în care se vor pune fișierele binare ale tuturor librăriilor utilizate, în funcție de configurația pentru care au fost compilate (ex. toate fișierele din folderul *Debug* din Figura 13 vor fi copiate în folderul „Debug”, iar toate fișierele din folderul *Release* din Figura 13 vor fi copiate în folderul „Release”). Puteți observa structura recomandată în Figura 14, iar Figura 15 prezintă structura folderului după adăugarea fișierelor bibliotecii GLFW.



Figura 14 - Structura recomandată

OpenGL dev libs	Name	Date modified	Type
include	glfw3.h	20.01.2020 00:43	C/C++ Header
GLFW	glfw3native.h	20.01.2020 00:43	C/C++ Header
lib			
Debug			
Release			

OpenGL dev libs	Name	Date modified	Type
include			
GLFW	glfw3.lib	22.09.2020 16:58	Object File Library
lib			
Debug			
Release			

OpenGL dev libs	Name	Date modified	Type
include			
GLFW	glfw3.lib	22.09.2020 16:58	Object File Library
lib			
Debug			
Release			

Figura 15 - Structura folderului după adăugarea fișierelor bibliotecii GLFW

2. Biblioteca GLEW

Similar cu biblioteca GLM, pentru a putea dezvolta o aplicație folosind biblioteca GLEW este nevoie să se obțină fișierele header (extensia .h) ale bibliotecii și fișierele binare (extensia .lib) ale acesteia.

Metoda recomandată pentru a obține fișierele binare este de a compila întreaga bibliotecă din surse. Doar compilarea din surse asigură compatibilitate maximă între bibliotecă și platforma pe care se vor dezvolta aplicațiile. Alternativ, binarele precompilate pentru diferite platforme predefinite pot fi descărcate de pe website-ul <http://glew.sourceforge.net/>.

2.1. Compilarea bibliotecii din surse

- Sursele bibliotecii pot fi descărcate de pe website-ul <http://glew.sourceforge.net/>. (Figura 16)

The OpenGL Extension Wrangler Library

The OpenGL Extension Wrangler Library (GLEW) is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. OpenGL core and extension functionality is exposed in a single header file. GLEW has been tested on a variety of operating systems, including Windows, Linux, Mac OS X, FreeBSD, Irix, and Solaris.

Downloads

GLEW is distributed as source and precompiled binaries.
The latest release is 2.1.0[07-31-17]:

Source	ZIP	TGZ
Binaries	Windows 32-bit and 64-bit	

Figura 16 - Descărcarea surselor bibliotecii GLEW

- Procesul de compilare a bibliotecii GLEW este identic cu cel folosit pentru biblioteca GLFW, cu mențiunea că fișierele necesare pentru CMake se găsesc în <folderul surselor> -> build -> cmake, ceea ce duce la căile ilustrate în Figura 17.

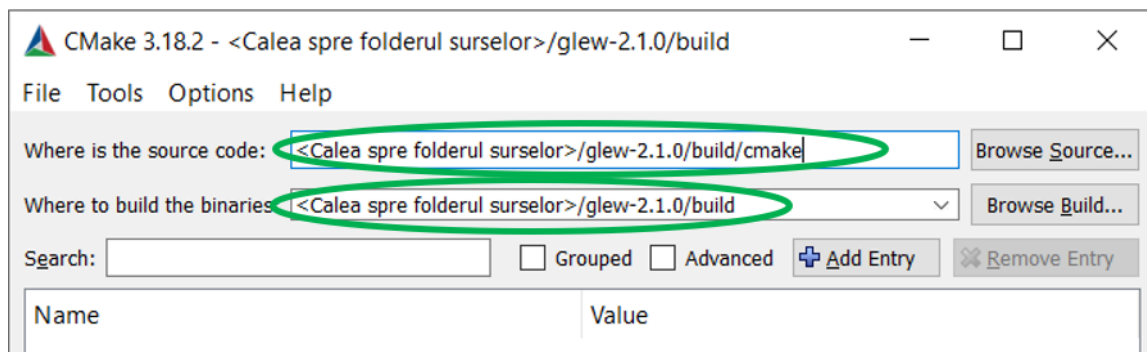


Figura 17 - Calea spre sursele bibliotecii GLEW

- Atenție! (doar pentru cei care folosesc MVS 2019)** Din cauza unor modificări pe care Microsoft Visual Studio 2019 le-a suferit față de variantele anterioare, pentru a putea configura și compila cu succes proiectul MVS al bibliotecii GLEW, este nevoie să editați înainte fișierul >folderul surselor>->build->cmake-> CMakeLists.txt, adăugând librăriile *vcruntime* în procesul de compilare (Figura 18) (exista un spatiu dupa *vcruntime*).

```

if (MSVC)
    # add options from visual studio project
    target_compile_definitions(glew PRIVATE "GLEW_BUILD;VC_EXTRALEAN")
    target_compile_definitions(glew_s PRIVATE "GLEW_STATIC;VC_EXTRALEAN")
    target_link_libraries(glew LINK_PRIVATE vcruntime -BASE:0x62AA0000)
    # kill security checks which are dependent on stdlib
    target_compile_options(glew PRIVATE -GS-)
    target_compile_options(glew_s PRIVATE -GS-)
    # remove stdlib dependency
    target_link_libraries(glew LINK_PRIVATE -nodefaultlib -noentry)
    string(REGEX REPLACE "/RTC(su|lsu)" "" CMAKE_C_FLAGS_DEBUG ${CMAKE_C_FLAGS_DEBUG})
elseif (WIN32 AND ((CMAKE_C_COMPILER_ID MATCHES "GNU") OR (CMAKE_C_COMPILER_ID MATCHES "Clang")))
    # remove stdlib dependency on windows with GCC and Clang (for similar reasons
    # as to MSVC - to allow it to be used with any Windows compiler)
    target_compile_options(glew PRIVATE -fno-builtin -fno-stack-protector)
    target_compile_options(glew_s PRIVATE -fno-builtin -fno-stack-protector)
    target_link_libraries(glew LINK_PRIVATE -nostdlib)
endif ()

```

Figura 18 - Modificarea fișierului CMakeLists.txt

- După compilare, fișierele header ale bibliotecii GLFW se vor găsi în folderul *include* din folderul rădăcină al bibliotecii (Figura 19), iar fișierele binare se vor găsi în folderul build->lib-> <numele configurației (Debug sau Release)> (Figura 21). Întrucât biblioteca GLEW se compilează implicit ca o librărie partajată (dinamică),
- Pentru utilizatorii avansați:** Dacă doriți să utilizați varianta dinamică (partajată) a bibliotecii, fișierul DLL se va găsi în folderul build->bin-><numele configurației (Debug sau Release)> (Figura 20).



Figura 19 - Calea spre fişierele header

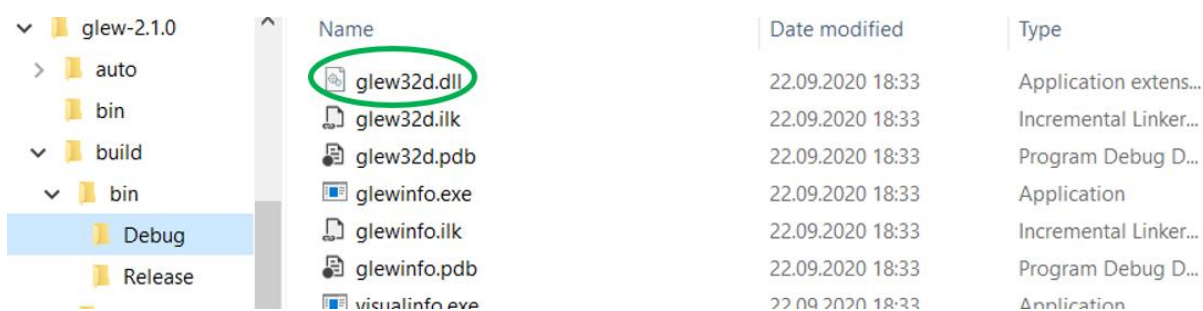


Figura 20 - Calea spre fişierele dll

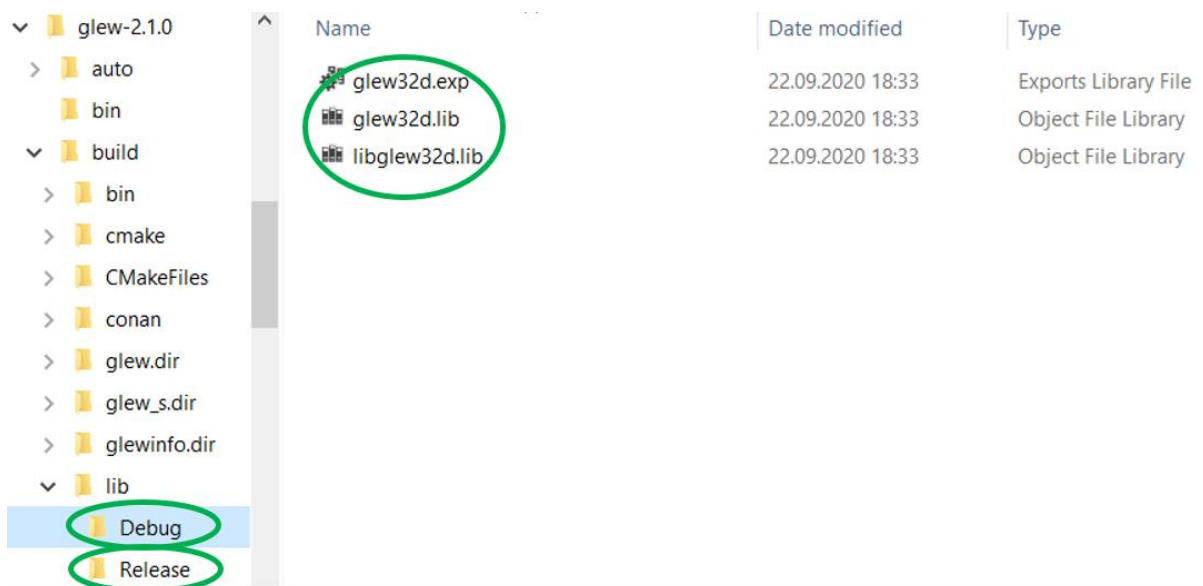


Figura 21 - Calea spre fişierele binare

2.1. Organizarea fişierelor bibliotecii GLEW

- Fişierele bibliotecii GLEW vor fi organizate conform convenţiei prezentate în secţiunea 1.2. Puteţi observa în Figura 22 structura recomandată după ce se vor adăuga fişierele GLEW.

OpenGL dev libs	Name	Date modified	Type
include	eglew.h	31.07.2017 14:25	C/C++ Header
GL	glew.h	31.07.2017 14:25	C/C++ Header
GLFW	glfw.h	31.07.2017 14:25	C/C++ Header
lib	wglew.h	31.07.2017 14:25	C/C++ Header

OpenGL dev libs	Name	Date modified	Type
include	glew32.dll	30.09.2020 16:35	Application extension
GL	glew32.exp	30.09.2020 16:35	Exports Library File
GLFW	glew32.lib	30.09.2020 16:35	Object File Library
lib	glfw3.lib	22.09.2020 16:58	Object File Library
Debug	libglew32.lib	30.09.2020 16:35	Object File Library
Release			

OpenGL dev libs	Name	Date modified	Type
include	glew32.dll	30.09.2020 16:36	Application extension
GL	glew32.exp	30.09.2020 16:36	Exports Library File
GLFW	glew32.lib	30.09.2020 16:36	Object File Library
lib	glfw3.lib	22.09.2020 16:58	Object File Library
Debug	libglew32.lib	30.09.2020 16:36	Object File Library
Release			

Figura 22 - Structura folderului după adăugarea fișierelor bibliotecii GLEW

3. Microsoft Visual Studio

3.1. Instalarea Microsoft Visual Studio

- Microsoft Visual Studio Community poate fi descărcat gratuit de pe website-ul <https://visualstudio.microsoft.com/vs/community/> (Figura 23). Acesta necesită doar un cont Microsoft pentru a putea fi folosit fără întrerupere.

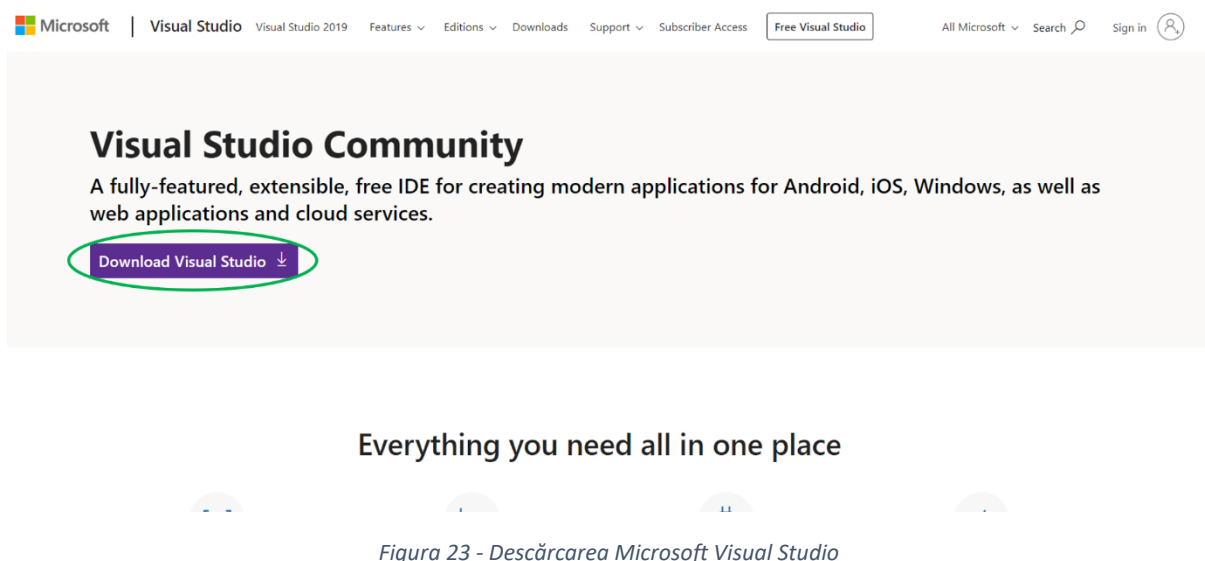


Figura 23 - Descărcarea Microsoft Visual Studio

- Modulul necesar și suficient pentru laborator este cel de dezvoltare aplicații desktop folosind C++ (Figura 24)

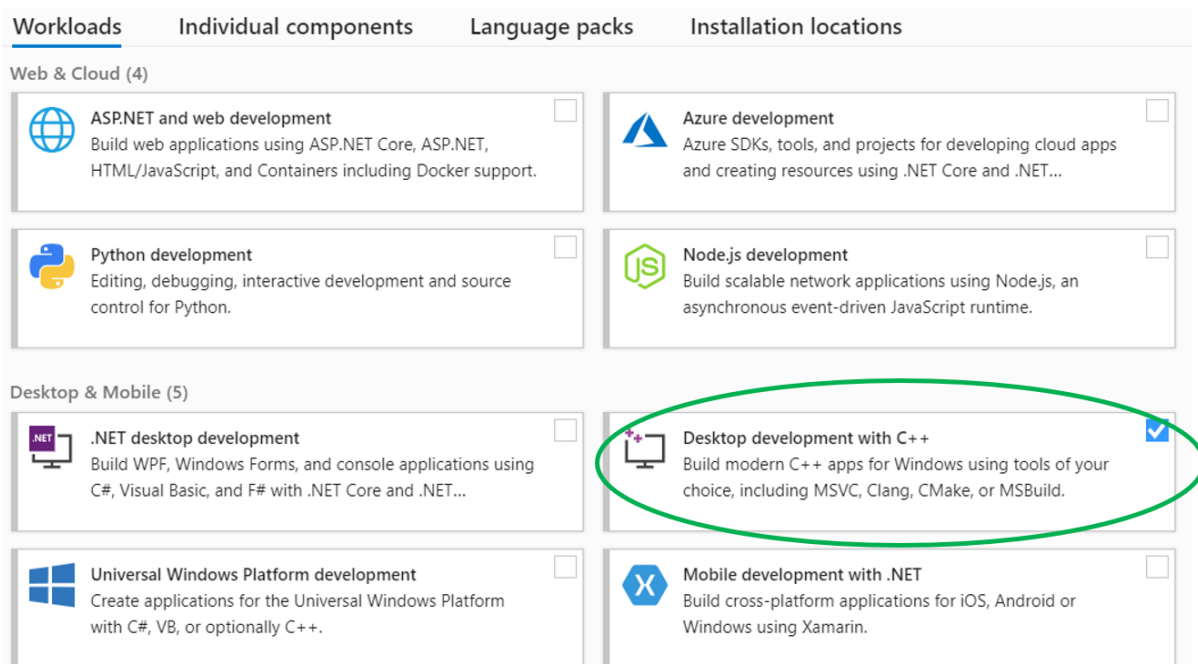


Figura 24 - Selectarea modulelor MVS

3.2. Crearea unui proiect nou OpenGL în Microsoft Visual Studio

- Creați un nou proiect Visual Studio (Figura 25).

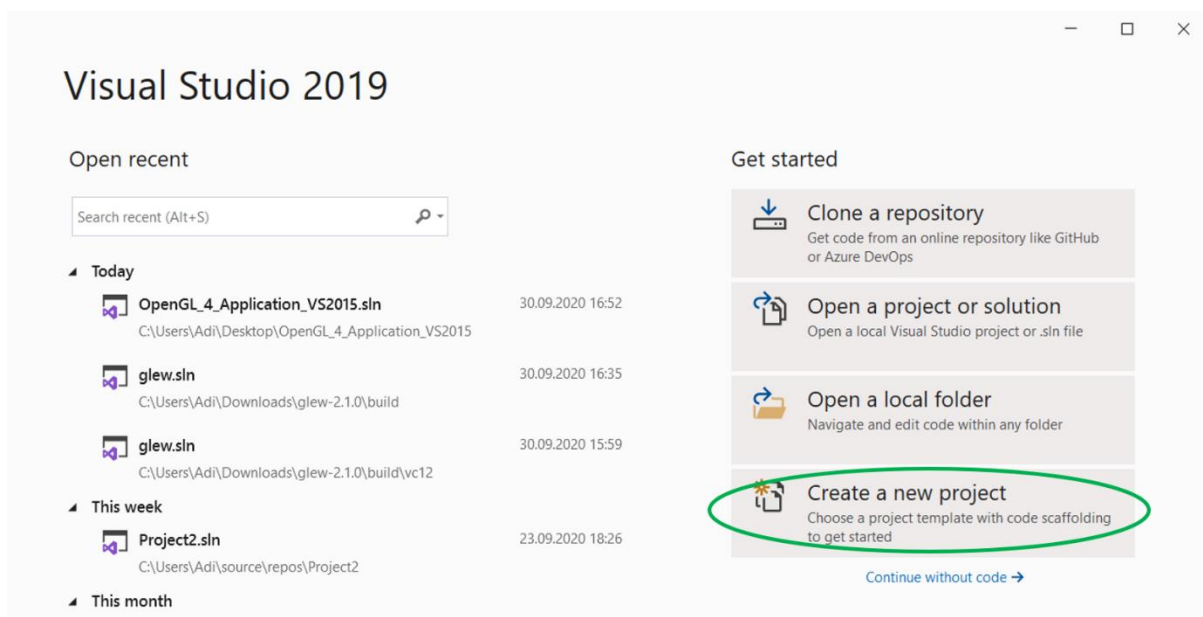


Figura 25 - Crearea unui proiect nou

- Utilizați șablonul „Empty Project” ().

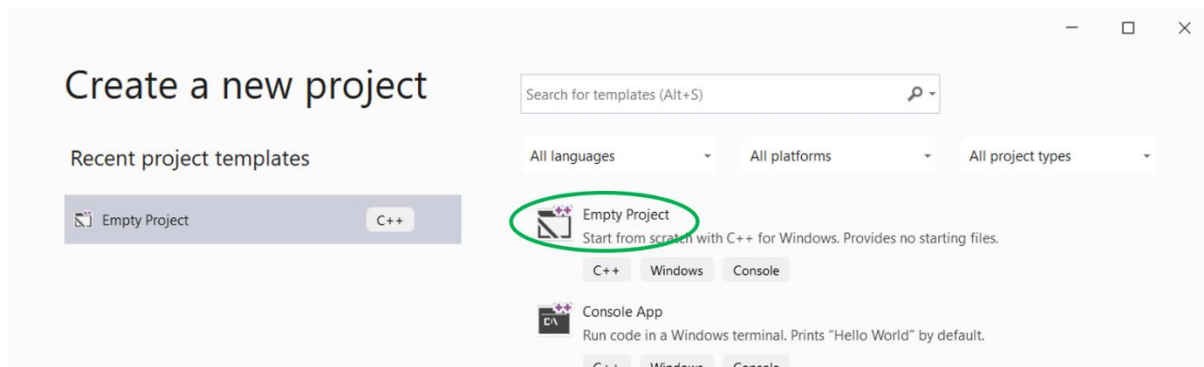


Figura 26 - Alegerea șablonului

- Copiați folderul „OpenGL dev libs” în folderul rădăcină al proiectului curent. Pentru a deschide folderul rădăcină, Efectuați click-dreapta pe numele proiectului și selectați opțiunea „Open Folder in File Explorer” (Figura 27)

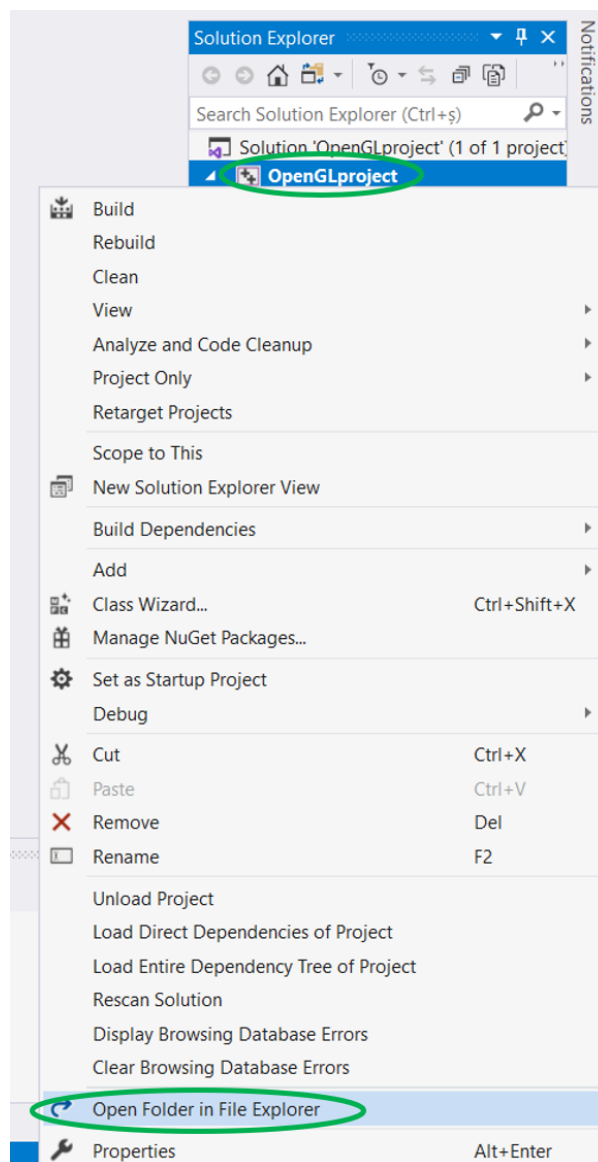


Figura 27 - Accesarea folderului proiectului

- Adăugați calea spre fișierele header ale librăriilor auxiliare în *Include Path*
 - Selectați din meniu Project->Properties (Figura 28)

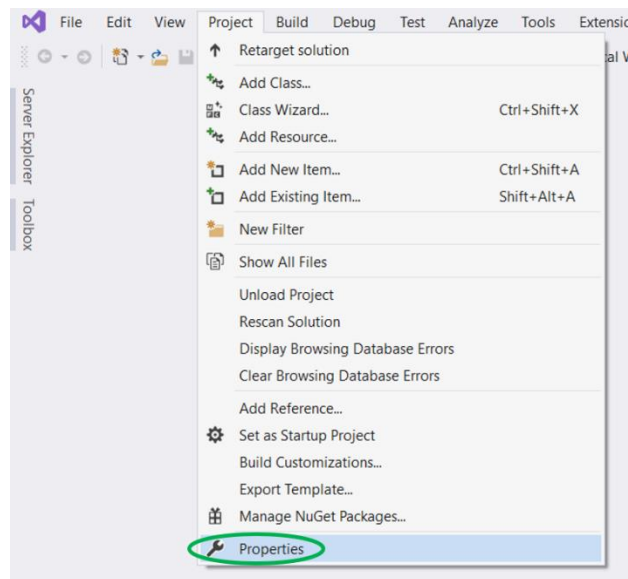


Figura 28 - Accesarea proprietăților proiectului

- Adăugați calea spre folderul ce conține fișierele header ale librăriilor auxiliare la rubrica „Additional Include Directories” din Proprietățile „C/C++”, secțiunea „General”. Asigurați-vă că faceți modificările pentru **toate configurațiile** (atât Debug cât și Release) (Figura 29 și Figura 30). Pentru a va apărea Proprietățile „C/C++”, deschideți sau creați o sursă cu extensia .cpp în proiectul curent. Dacă nu apar Proprietățile „C/C++”, selectați Proprietățile „VC++ Directories”, și scrieți calea la rubrica „Include Directories”. **Atenție!** Adăugați sursele din folderul „root” al proiectului, nu alte copii ale acestora existente în alte locații.

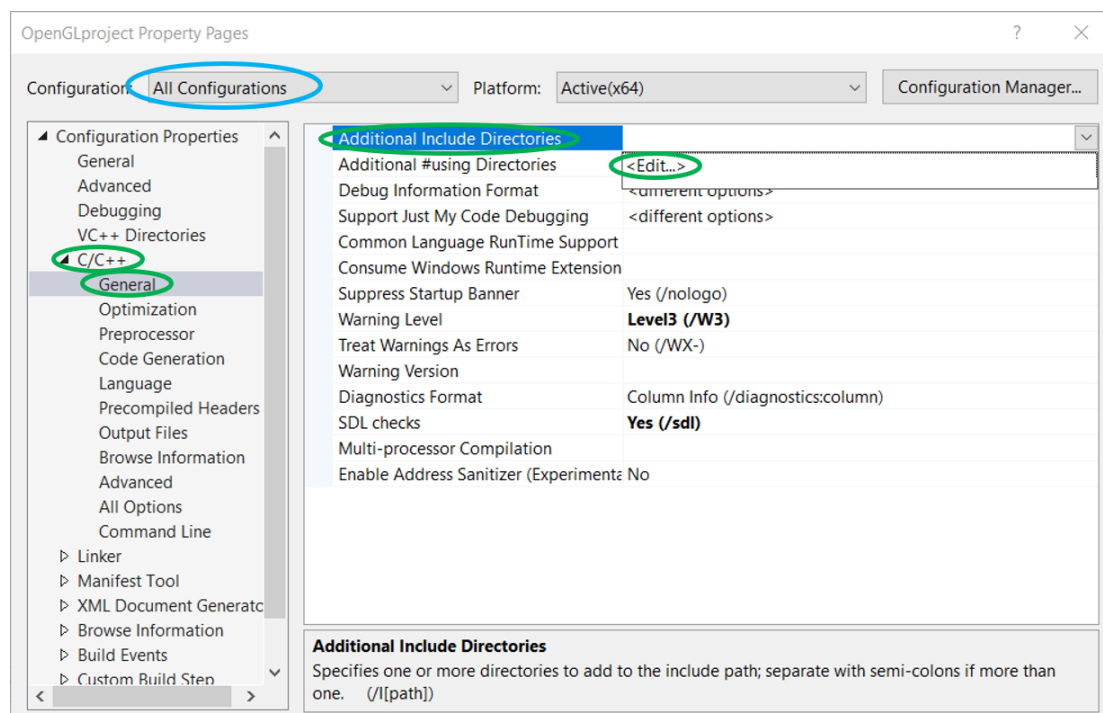


Figura 29 - Adăugarea folderelor cu fișiere header suplimentare

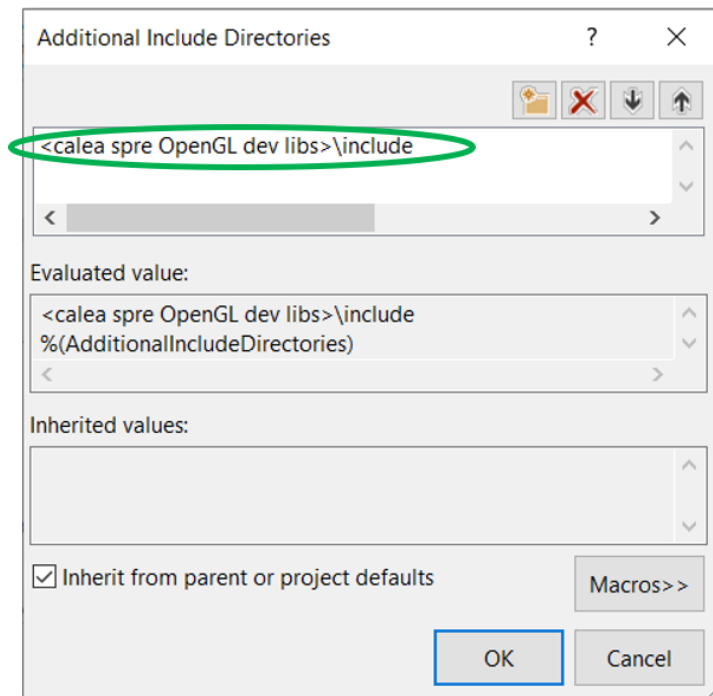


Figura 30 - Calea spre fișierele header

- Adăugați calea spre fișierele binare ale librăriilor auxiliare în *Library Path*
 - Adăugați calea spre folderul ce conține fișierele binare ale librăriilor auxiliare la rubrica „Additional Library Directories” din Proprietățile „Linker”, secțiunea „General”. Asigurați-vă că pentru configurația Debug adăugați calea spre folderul care conține varianta Debug a binarelor (<calea spre OpenGL dev libs>\lib\Debug), iar pentru configurația Release adăugați calea spre folderul care conține varianta Release a binarelor (<calea spre OpenGL dev libs>\lib\Release) (Figura 31)

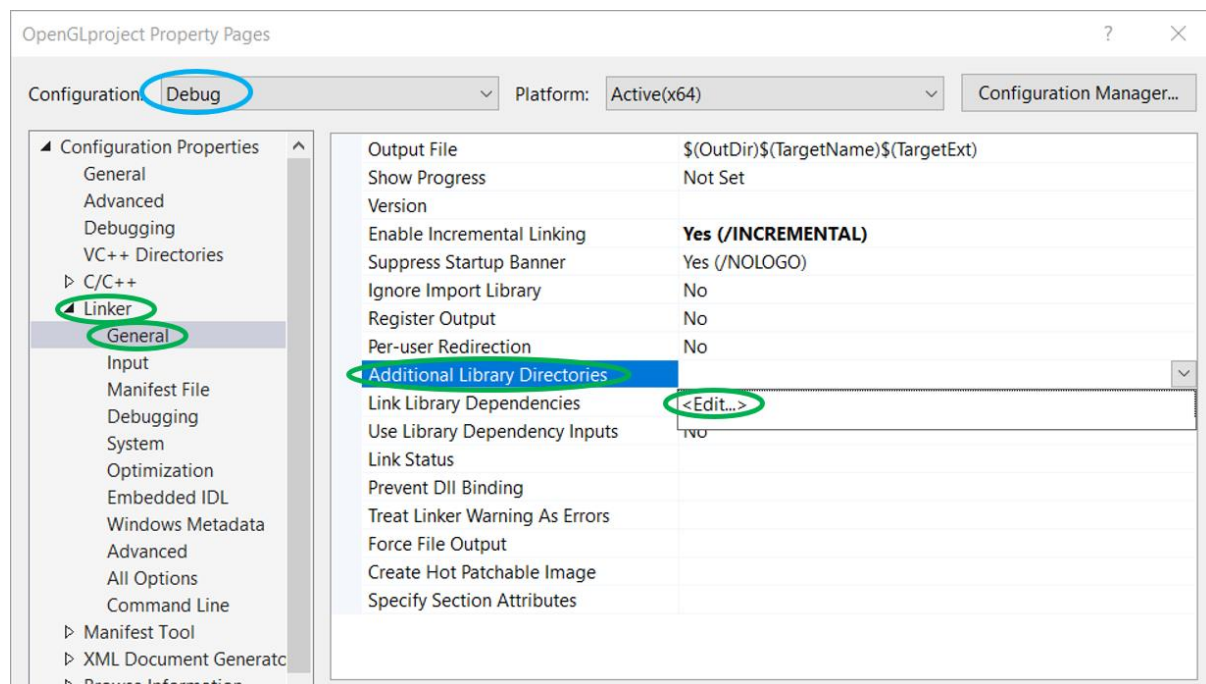


Figura 31 - Adăugarea folderelor cu fișiere binare suplimentare

- Adăugați numele librărilor auxiliare care vor fi folosite la rubrica „Additional Dependencies” din Proprietățile „Linker”, secțiunea „Input” (Figura 32). **Atenție!** Se poate întâmpla ca numele binarelor să difere în funcție de configurația pentru care au fost create (Debug sau Release). Se recomandă consultarea numelor fișierelor binare din folderul cu librării auxiliare înainte de adăugarea lor pentru cele două configurații.

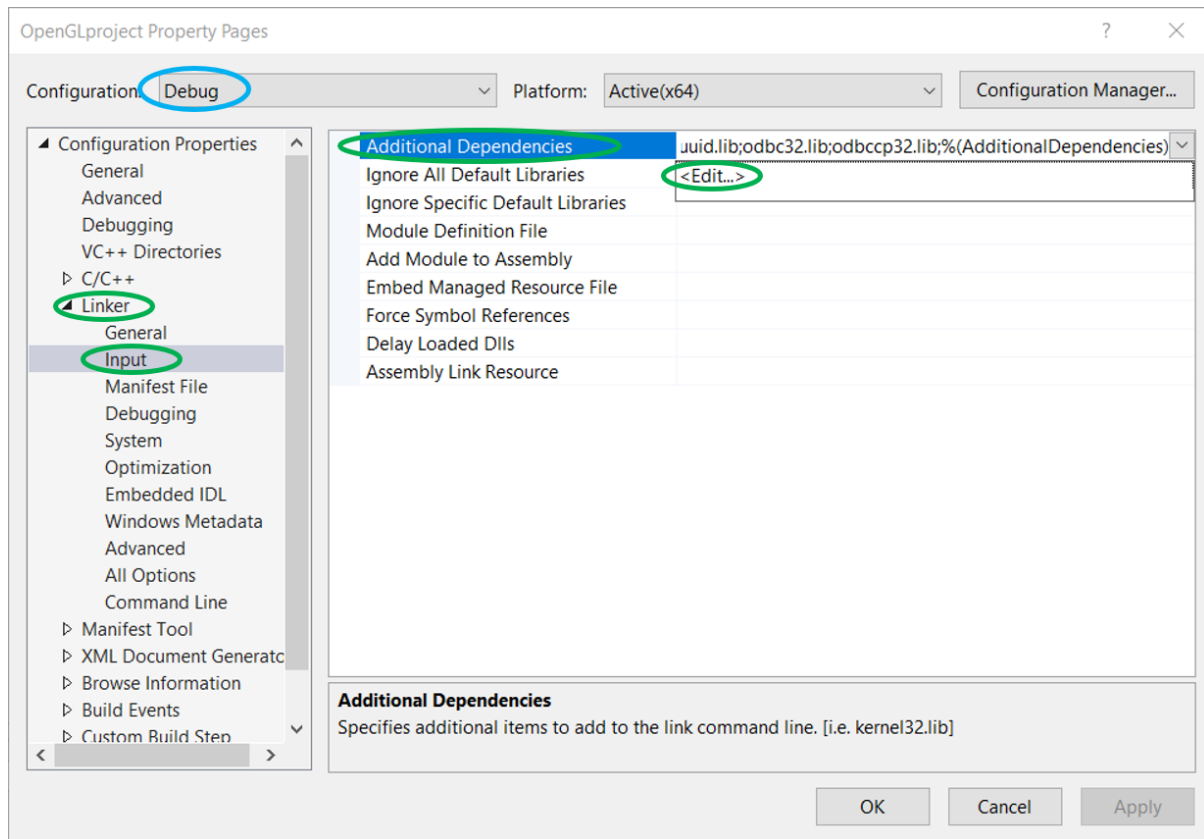


Figura 32 - Adăugarea numelor librărilor

Numele librărilor adiționale necesare sunt ilustrate în Figura 33 Configurația Debug (opengl32.lib glfw3.lib libglew32d.lib) în stânga și configurația Release (opengl32.lib glfw3.lib libglew32.lib) în dreapta. Cu toate că unele dependențe nu apar în calea fișierelor, ele trebuie scrise deoarece sunt împachetate și incluse în celelalte dependențe.

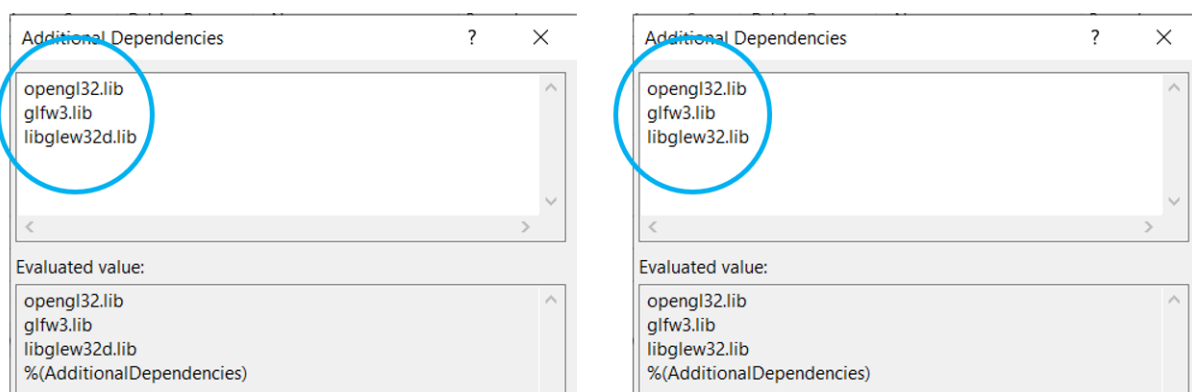


Figura 33 - Numele librărilor adiționale

3.3. Crearea unui nou șablon de proiect pe baza proiectului complet configurat

- După configurarea completă a proiectului, acesta poate fi folosit ca baza unui nou șablon pentru Microsoft Visual Studio
- Selectați din meniu Project->Export Template (Figura 34)

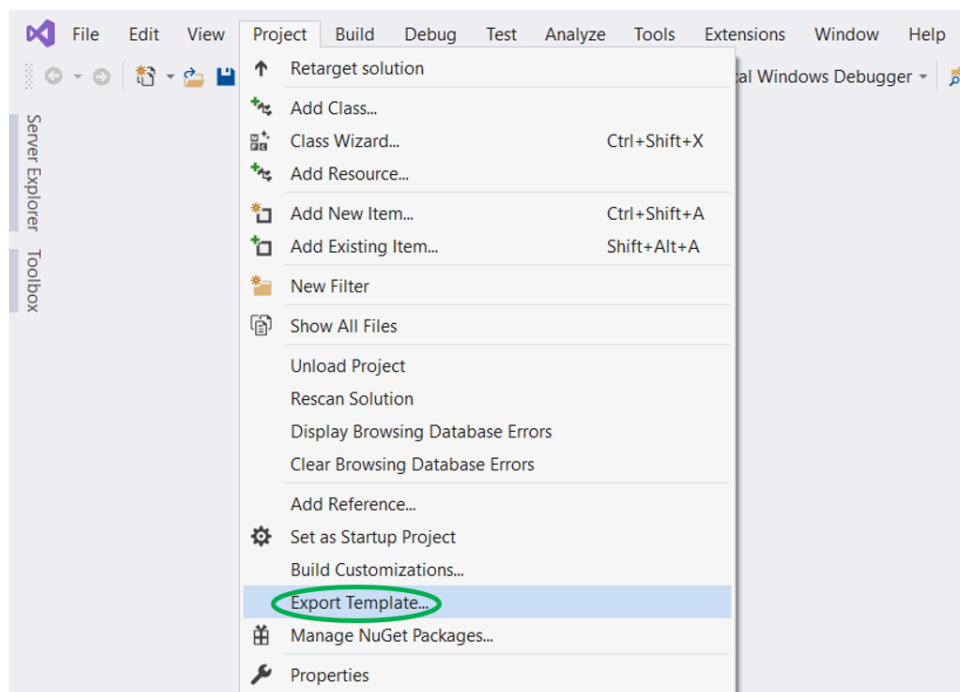


Figura 34 - Exportarea proiectului ca șablon

- Selectați opțiunea „Project template” (Figura 35)

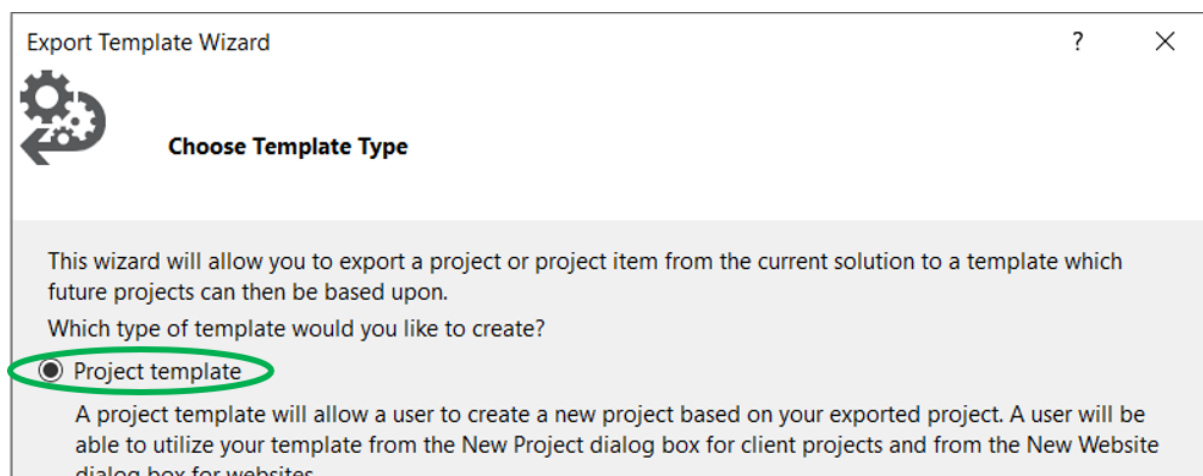
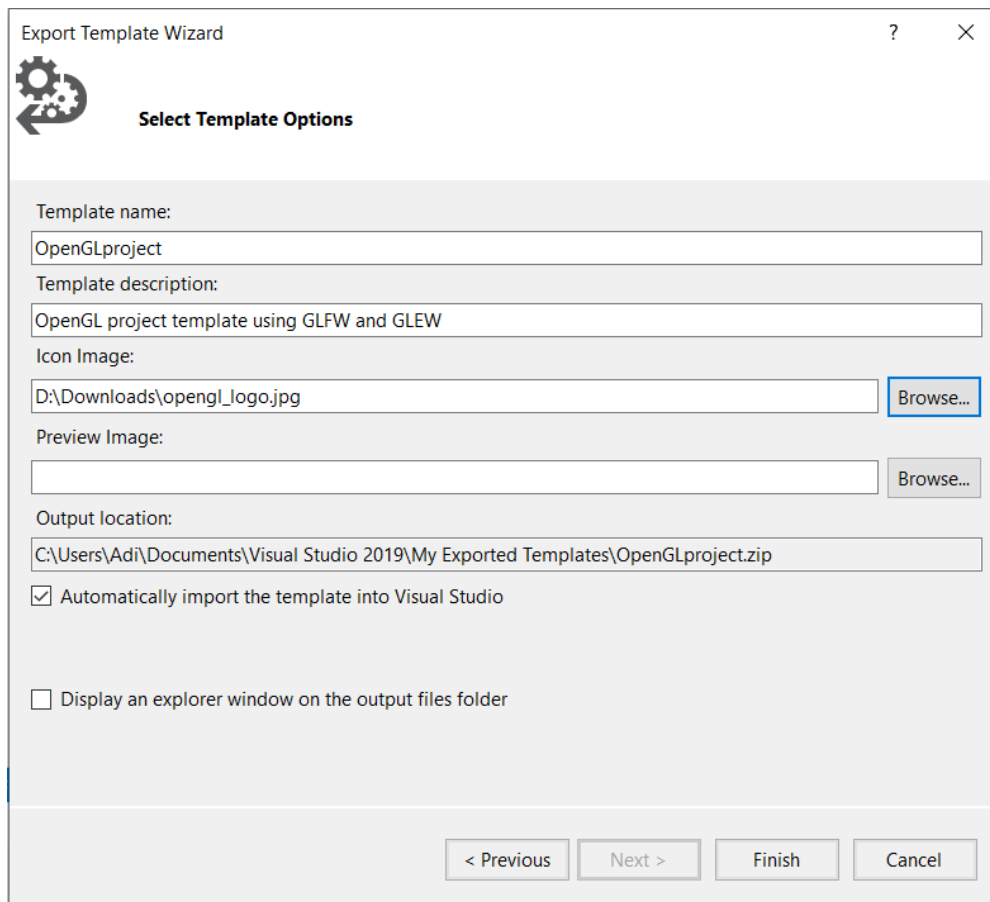


Figura 35 - Tipul sablonului

- Completați câmpurile descriptive ale șablonului (Figura 36)



Export Template Wizard

Select Template Options

Template name:
OpenGLproject

Template description:
OpenGL project template using GLFW and GLEW

Icon Image:
D:\Downloads\opengl_logo.jpg Browse...

Preview Image:
Browse...

Output location:
C:\Users\Adi\Documents\Visual Studio 2019\My Exported Templates\OpenGLproject.zip

☒ Automatically import the template into Visual Studio

☐ Display an explorer window on the output files folder

< Previous Next > Finish Cancel

Figura 36 - Câmpurile descriptive ale șablonului

- Șablonul poate fi acum utilizat pentru crearea unui proiect nou (Figura 37)

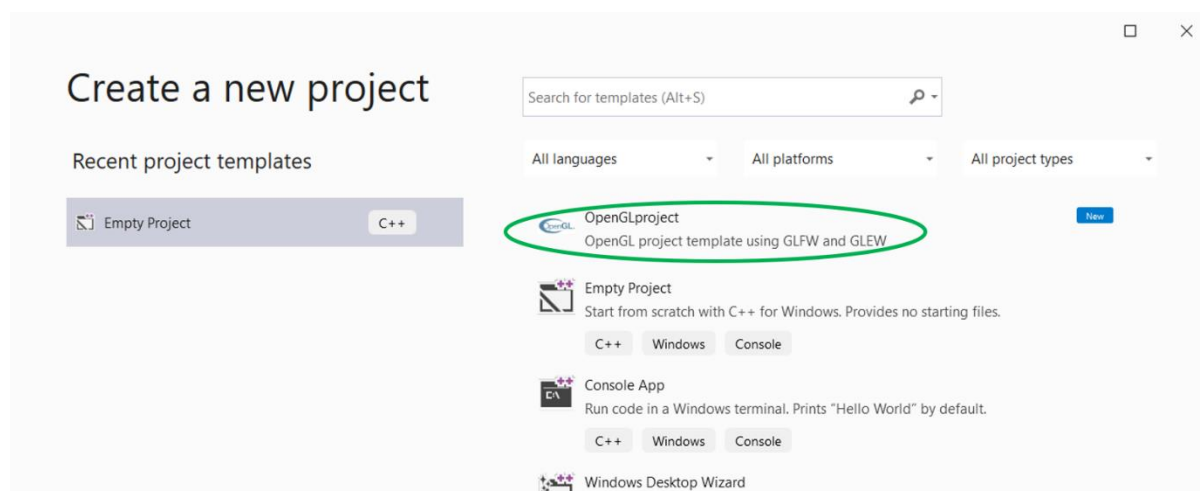


Figura 37 - Utilizarea șablonului