Build & Run nesting2

Prerequisites

You need to have:

- a Windows system
- Git installed (see how here: https://git-scm.com/download/win)
- Cmake installed (see how here: https://cmake.org/download/)
- 7zip installed (https://www.7-zip.org/)

What you'll do

You will download and build the following dependencies:

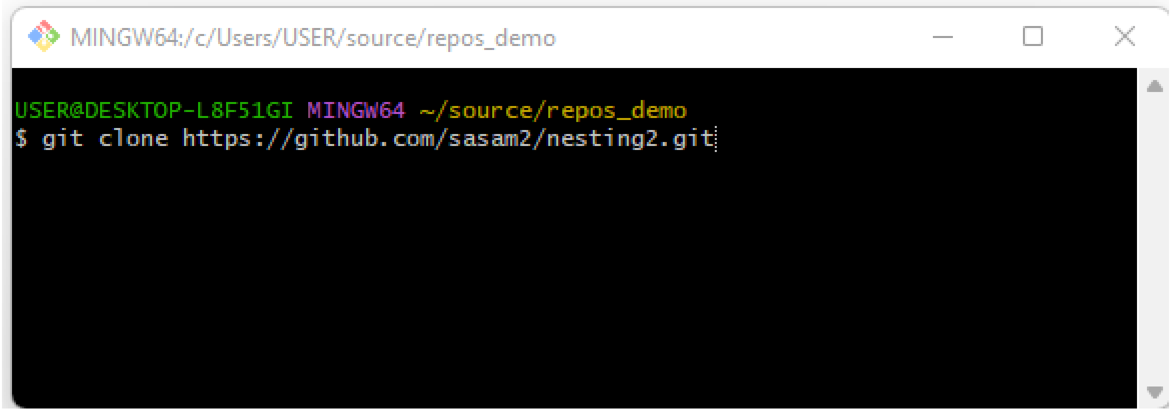
- boost (https://www.boost.org)
- CGAL (https://www.cgal.org/)
- freeglut (https://freeglut.sourceforge.net/)
- glew (https://glew.sourceforge.net/)
- glui (https://github.com/libglui/glui)
- opencv (https://opencv.org/)

After that you'll update the dependencies references on the Visual Studio project of nesting2 and you will build and run the project..

Setting up

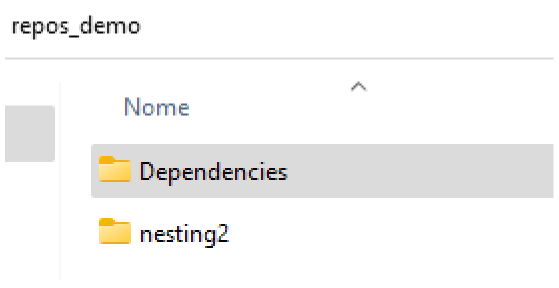
1.Create or select a parent folder where to clone your repo into. Mine is "repos_demo":

2. Clone the nesting2 project into your parent folder:



```
USER@DESKTOP-L8F51GI MINGW64 ~/source/repos_demo
$ git clone https://github.com/sasam2/nesting2.git
```

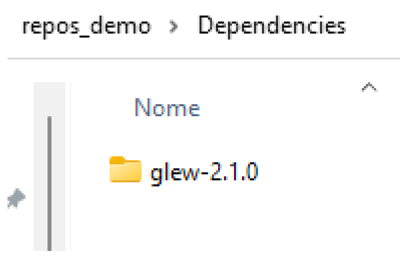
3. Create a folder called “Dependencies” into your parent folder (repos_demo, in this example).



Getting and building the dependencies

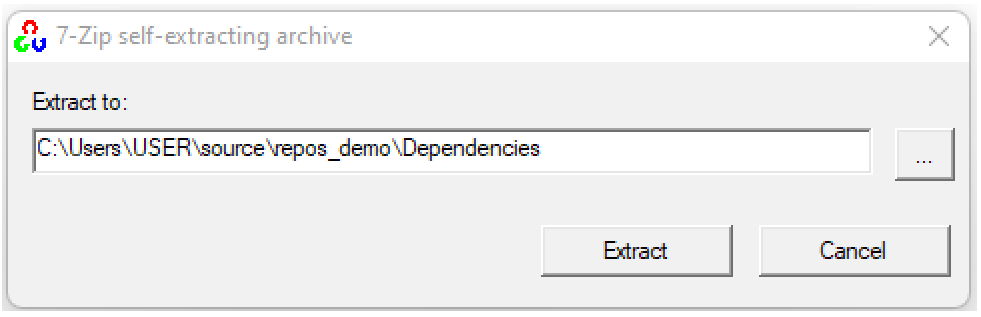
1. Get Glew

Go to https://glew.sourceforge.net/ and download the Windows binaries as a zip. Then extract the zip. The contents of the zip consist of a folder called “glew-2.1.0” (as of now, the version numbers can change). Copy that folder into the Dependencies folder.



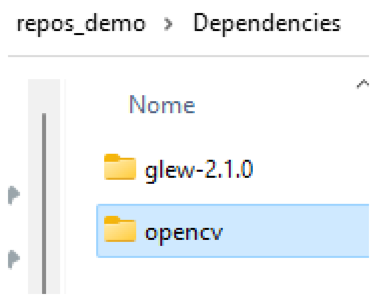
2. Get OpenCV

Go to https://opencv.org/releases/ and download the desired version (4.8.0 as of now) for windows. Once downloaded run the executable. You will be prompted with the destination folder you want to extract the contents to – choose the Dependencies folder.



C:\Users\USER\source\repos_demo\Dependencies

In the end you should also have a folder called “opencv” in your Dependencies folder.


repos_demo › Dependencies

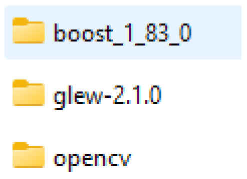
Nome

glew-2.1.0

opencv

3. Get Boost

Go to https://www.boost.org/users/download/ and download the desired version (1.83.0 as of now) for windows. Then extract the zip. The contents of the zip consist of a folder called “boost_1_83_0” . Copy that folder into the Dependencies folder.

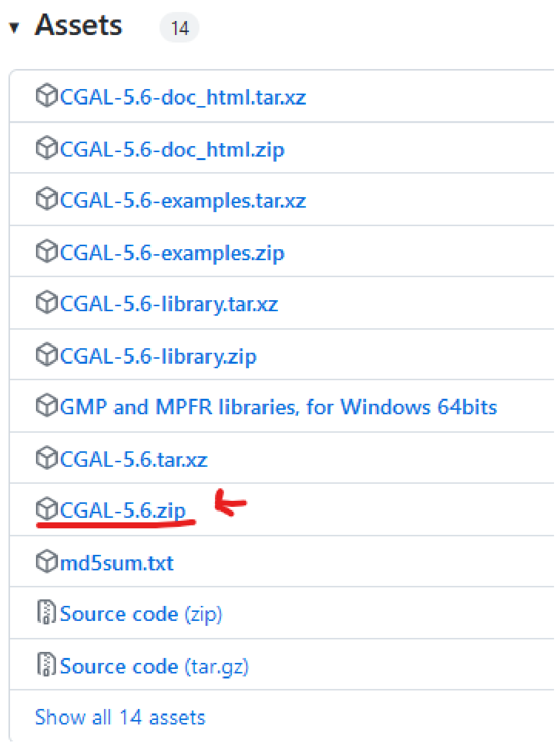

boost_1_83_0

glew-2.1.0

opencv

4. Get CGAL

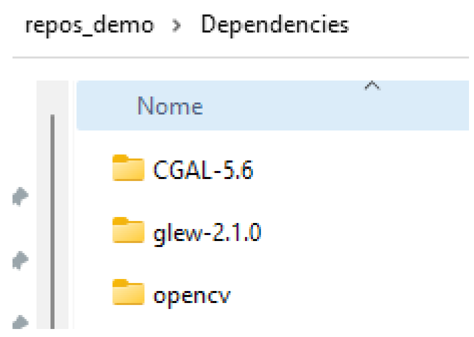
Go to https://github.com/CGAL/cgal/releases and pick the desired one (5.6 as of now) and download the zip file:


▾ Assets 14

- CGAL-5.6-doc_html.tar.xz
- CGAL-5.6-doc_html.zip
- CGAL-5.6-examples.tar.xz
- CGAL-5.6-examples.zip
- CGAL-5.6-library.tar.xz
- CGAL-5.6-library.zip
- GMP and MPFR libraries, for Windows 64bits
- CGAL-5.6.tar.xz
- CGAL-5.6.zip
- md5sum.txt
- Source code (zip)
- Source code (tar.gz)

Show all 14 assets

Then extract the zip. The contents of the zip consist of a folder called “CGAL-5.6”. Copy that folder into the Dependencies folder.



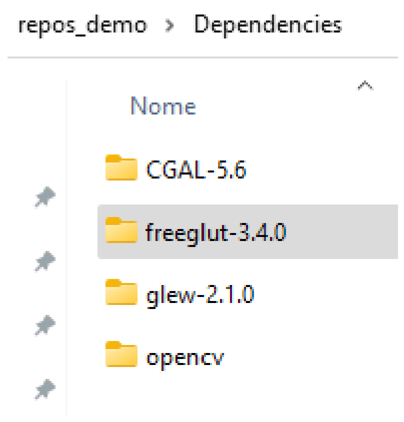
CGAL-5.6
glew-2.1.0
opencv

(Note: there is also the option of installing this dependency using **vcpkg**, but I’ve never tried it so I can’t recommend it).

5. Get and build Freeglut

Go to https://freeglut.sourceforge.net/ and download the desired version (3.4.0 as of now).

Unpack the .tar.gz file. The contents of this package consist of a folder called “freeglut-3.4.0”. Copy that folder into the Dependencies folder.

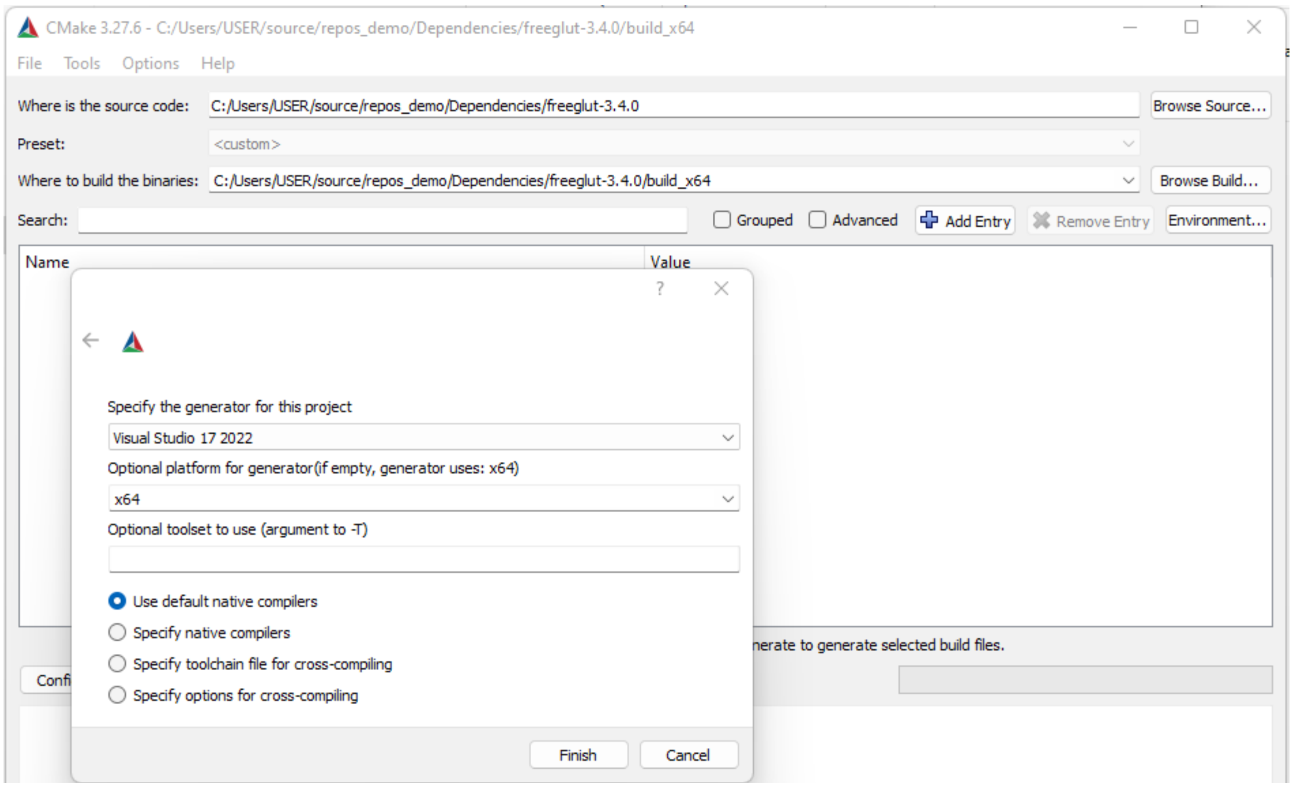


CGAL-5.6
freeglut-3.4.0
glew-2.1.0
opencv

At this point we have the sources of freeglut. Now we will use CMake to generate a Visual Studio solution that allows us to build the binaries.

Launch CMake.
Fill in the field “Where is the source code” with:
C:/<your-path-here>/repos_demo/Dependencies/freeglut-3.4.0
Fill in the field “Where to build the binaries” with:
C:/<your-path-here>/repos_demo/Dependencies/freeglut-3.4.0/build_x64
Click “Configure”.
You will be prompted if you want to create the nonexistent folder **build_x64**, click yes.
You will be prompted to specify the generator for the project, choose your Visual Studio version (that should appear in the drop down box). You will also have to choose the platform generator, choose preferably **x64**.

Your configuration should look like the picture below:


CMake 3.27.6 - C:/Users/USER/source/repos_demo/Dependencies/freeglut-3.4.0/build_x64

Where is the source code: C:/Users/USER/source/repos_demo/Dependencies/freeglut-3.4.0

Preset: <custom>

Where to build the binaries: C:/Users/USER/source/repos_demo/Dependencies/freeglut-3.4.0/build_x64

Specify the generator for this project: Visual Studio 17 2022

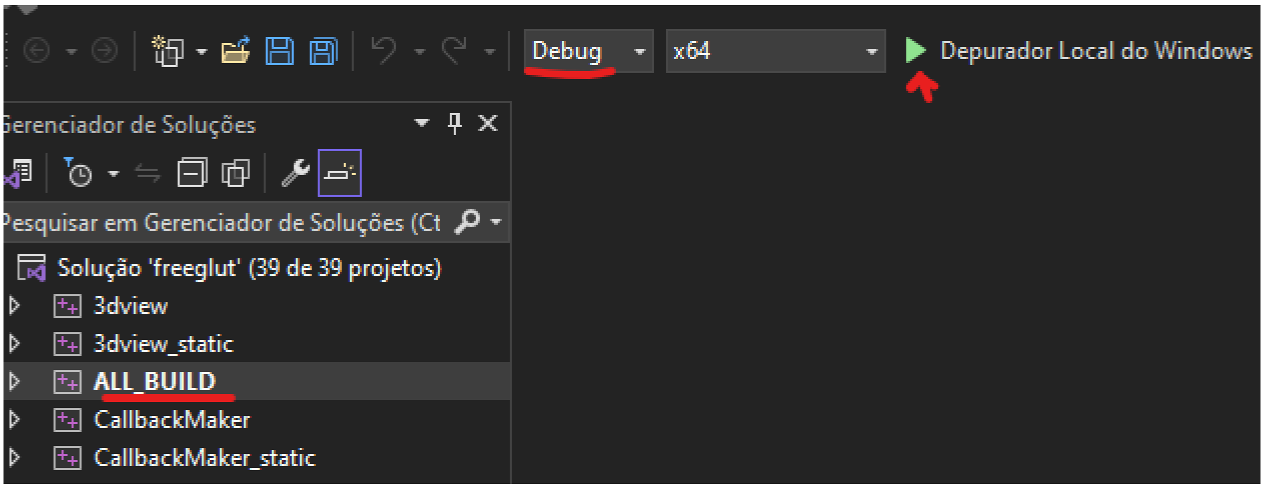
Optional platform for generator(if empty, generator uses: x64): x64

Optional toolset to use (argument to -T):

[✓ Use default native compilers]
Specify native compilers
Specify toolchain file for cross-compiling
Specify options for cross-compiling

Click “Configure” again to make the variables colored in red, white. And then click “Generate”. This generates the freeglut Visual Studio solution into the folder **repos_demo/ Dependencies/freeglut-3.4.0/build_x64.** Go to this folder and open the solution called **freeglut.sln**.

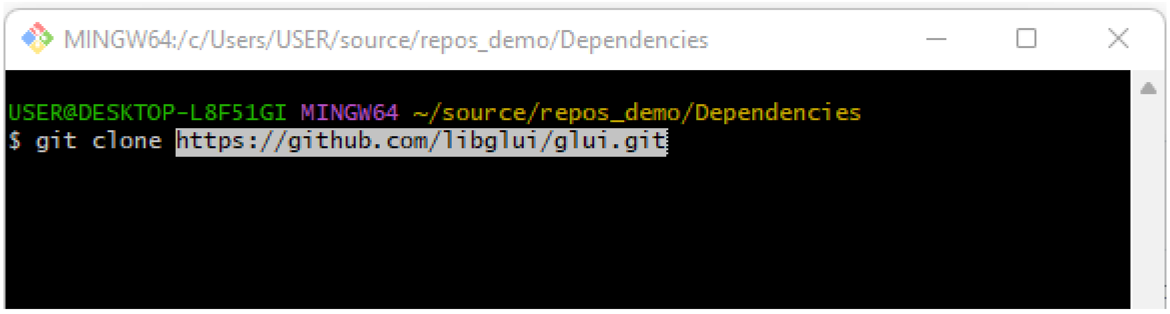
Run the **ALL_BUILD** in **debug** and **release** modes**.**


Debug
x64
Depurador Local do Windows
Gerenciador de Soluções
Pesquisar em Gerenciador de Soluções (Ct
Solução 'freeglut' (39 de 39 projetos)
3dview
3dview_static
ALL_BUILD
CallbackMaker
CallbackMaker_static

Now the libraries and the examples of code are all built.
The libraries are in
Dependencies\freeglut-3.4.0\build_x64\lib
The dll files are in
Dependencies\freeglut-3.4.0\build_x64\bin

6. Get and build Glui

Go to https://github.com/libglui/glui, and clone the repo into Dependencies folder.



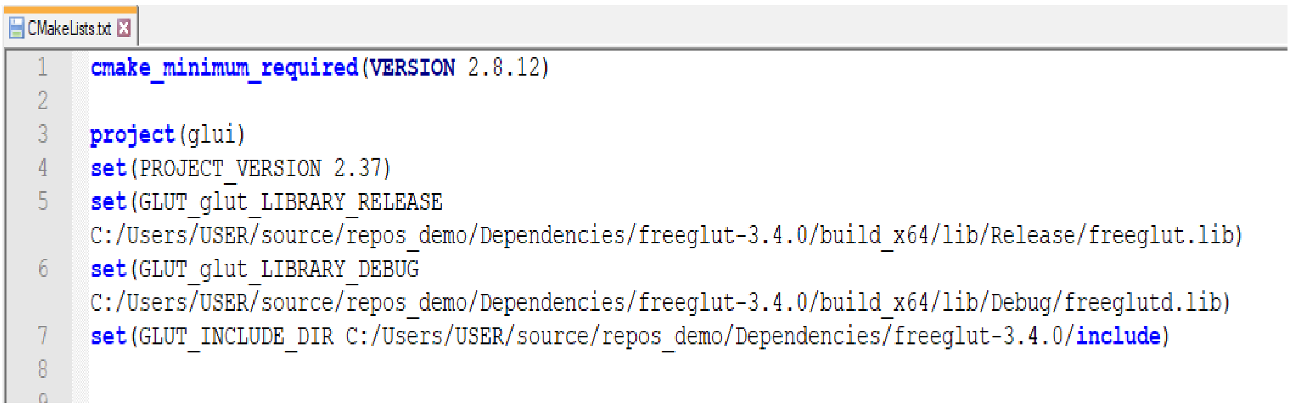
```
USER@DESKTOP-L8F51GI MINGW64 ~/source/repos_demo/Dependencies
$ git clone https://github.com/libglui/glui.git
```

At this point we have the sources of glui. Now we will use CMake to generate a Visual Studio solution that allows us to build the binaries.

Before you launch CMake, you need to tell CMake where freeglut library is, because it is needed to build glui. To do that, edit the CmakeLists file (in the root of glui folder) and add there the following three entries:

set(**GLUT_glut_LIBRARY_RELEASE**
C:/<your-path-here>/repos_demo/Dependencies/freeglut-3.4.0/build_x64/lib/Release/
freeglut.lib)
set(**GLUT_glut_LIBRARY_DEBUG**
C:/<your-path-here>/repos_demo/Dependencies
/freeglut-3.4.0/build_x64/lib/Debug/freeglutd.lib)
set(**GLUT_INCLUDE_DIR** .C:/<your-path-here>/repos_demo/Dependencies /freeglut-3.4.0/include)

This is what the file should look like:



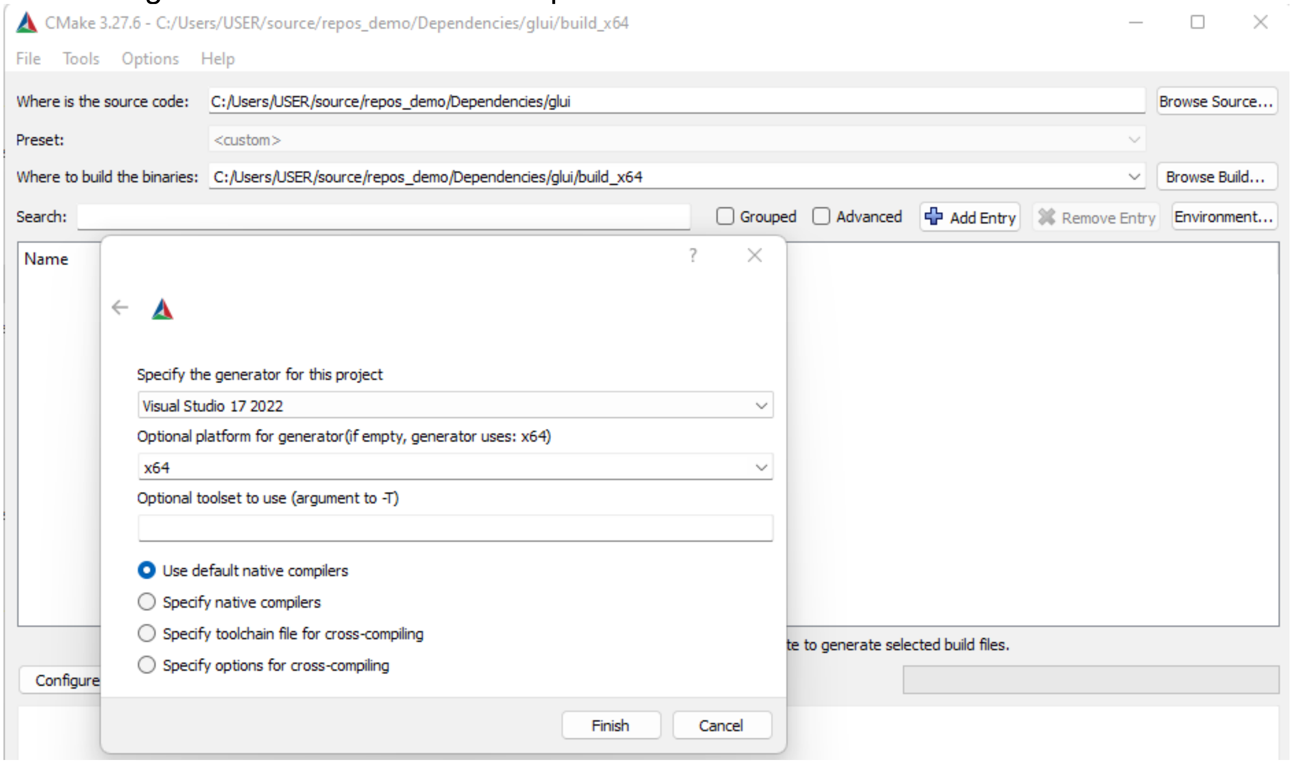
```
cmake_minimum_required(VERSION 2.8.12)

project(glui)
set(PROJECT_VERSION 2.37)
set(GLUT_glut_LIBRARY_RELEASE
C:/Users/USER/source/repos_demo/Dependencies/freeglut-3.4.0/build_x64/lib/Release/freeglut.lib)
set(GLUT_glut_LIBRARY_DEBUG
C:/Users/USER/source/repos_demo/Dependencies/freeglut-3.4.0/build_x64/lib/Debug/freeglutd.lib)
set(GLUT_INCLUDE_DIR C:/Users/USER/source/repos_demo/Dependencies/freeglut-3.4.0/include)
```

Launch CMake.
Fill in the field "Where is the source code" with:
C:/<your path here>/repos_demo/Dependencies/glui
Fill in the field "Where to build the binaries" with:
C:/<your path here>/repos_demo/Dependencies/glui/build_x64
Click "Configure".
You will be prompted if you want to create the nonexistent folder **build_x64**, click yes.

You will be prompted to specify the generator for the project, choose your Visual Studio version (that should appear in the drop down box). You will also have to choose the platform generator, choose preferably **x64**.

Your configuration should look like the picture below:



CMake 3.27.6 - C:/Users/USER/source/repos_demo/Dependencies/glui/build_x64

Where is the source code: C:/Users/USER/source/repos_demo/Dependencies/glui

Preset: <custom>

Where to build the binaries: C:/Users/USER/source/repos_demo/Dependencies/glui/build_x64

Specify the generator for this project: Visual Studio 17 2022

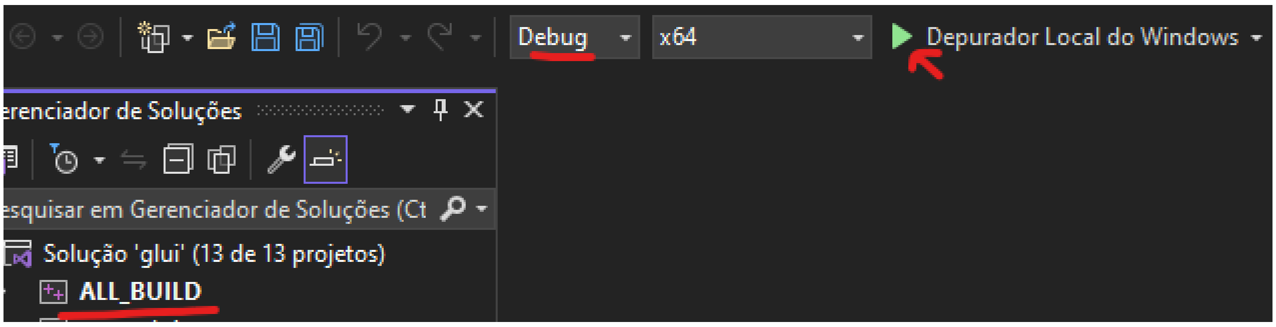
Optional platform for generator(if empty, generator uses: x64): x64

Optional toolset to use (argument to -T):

[✓ Use default native compilers]
Specify native compilers
Specify toolchain file for cross-compiling
Specify options for cross-compiling

Click “Configure” again to make the variables colored in red, white. And then click “Generate”. This generates the glui Visual Studio solution into the folder **repos_demo/Dependencies/glui/build_x64.** Go to this folder and open the solution called **glui.sln**.

Run the **ALL_BUILD** in **debug** and **release** modes**.**



Debug
x64
Depurador Local do Windows
erenciador de Soluções
esquisar em Gerenciador de Soluções (Ct
Solução 'glui' (13 de 13 projetos)
ALL_BUILD

Now the libraries and the examples of code are all built.
The binaries and libraries for debug are in
Dependencies\glui\build_x64\Debug
The binaries and libraries for release are in
Dependencies\glui\build_x64\Release

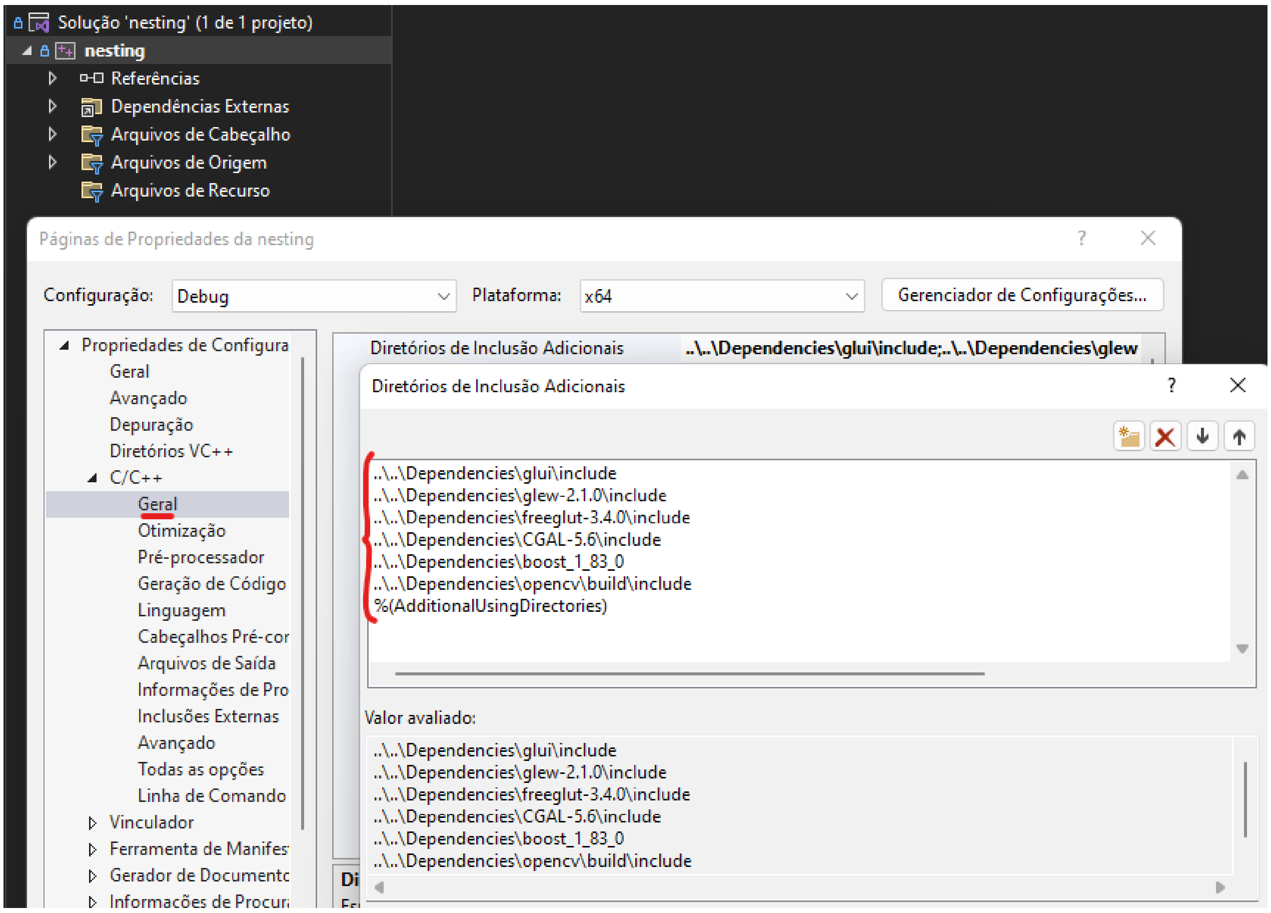
Build the nesting2 project

Now go to **repos_demo\nesting2** and open **nesting.sln** solution.

1. Verify that the paths to include folders are correct

Then right click on the project (called "nesting") and choose "Properties". On the left menu of the window that appears, expand the "C/C++" category and then choose the option "General". On the right frame you will find an entry called "Additional Include Directories", click on the arrow to edit this.

Your screen should look like the one below (sorry for not providing pictures in English):



Diretórios de Inclusão Adicionais ..\..\Dependencies\glui\include;..\..\Dependencies\glew

```
..\..\Dependencies\glui\include
..\..\Dependencies\glew-2.1.0\include
..\..\Dependencies\freeglut-3.4.0\include
..\..\Dependencies\CGAL-5.6\include
..\..\Dependencies\boost_1_83_0
..\..\Dependencies\opencv\build\include
%(AdditionalUsingDirectories)
```

Valor avaliado:

```
..\..\Dependencies\glui\include
..\..\Dependencies\glew-2.1.0\include
..\..\Dependencies\freeglut-3.4.0\include
..\..\Dependencies\CGAL-5.6\include
..\..\Dependencies\boost_1_83_0
..\..\Dependencies\opencv\build\include
```

You will find the relative paths to the "include" folders of the dependencies we are using. These will be:

..\..\Dependencies\glui\include

..\..\Dependencies\glew-2.1.0\include

..\..\Dependencies\freeglut-3.4.0\include

..\..\Dependencies\CGAL-5.6\include

..\..\Dependencies\boost_1_83_0

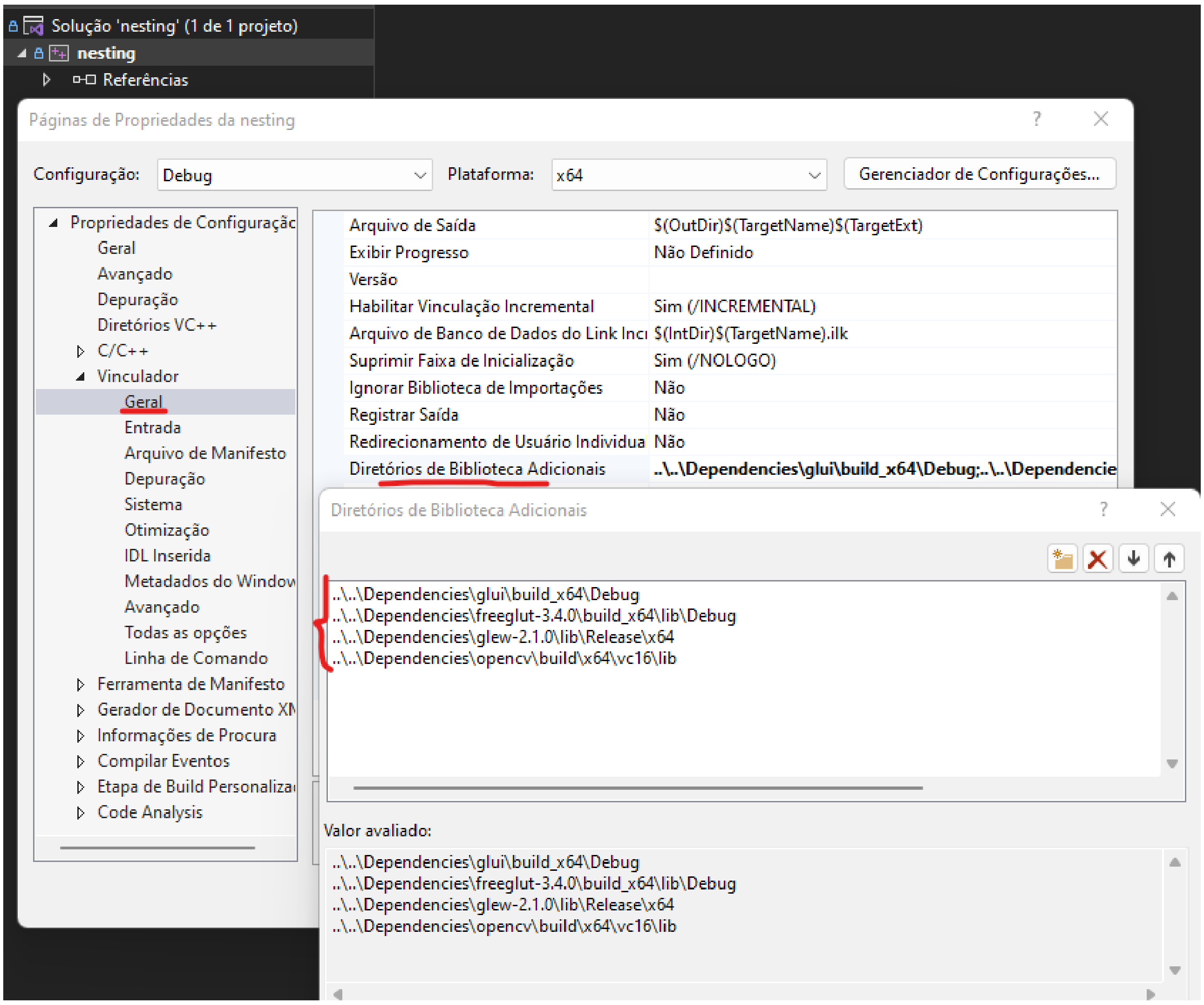
..\..\Dependencies\opencv\build\include

Confirm that these paths effectively correspond to the include folders provided by the libraries you downloaded. Note that, **if you downloaded different versions of the dependencies, you must change the paths to match your versions.**

2. Verify that the paths to libraries folders are correct

Go back to the properties window and expand the "Linker" category and choose the option "General". On the right frame you will find an entry called "Additional Library Directories" click on the arrow to edit this.

Your screen should look like this:



Configuração: Debug
Plataforma: x64

Arquivo de Saída	$(OutDir)$(TargetName)$(TargetExt)
Exibir Progresso	Não Definido
Versão	
Habilitar Vinculação Incremental	Sim (/INCREMENTAL)
Arquivo de Banco de Dados do Link Inc	$(IntDir)$(TargetName).ilk
Suprimir Faixa de Inicialização	Sim (/NOLOGO)
Ignorar Biblioteca de Importações	Não
Registrar Saída	Não
Redirecionamento de Usuário Individua	Não
Diretórios de Biblioteca Adicionais	**..\..\Dependencies\glui\build_x64\Debug;..\..\Dependencie**

```
..\..\Dependencies\glui\build_x64\Debug
..\..\Dependencies\freeglut-3.4.0\build_x64\lib\Debug
..\..\Dependencies\glew-2.1.0\lib\Release\x64
..\..\Dependencies\opencv\build\x64\vc16\lib
```

Valor avaliado:

```
..\..\Dependencies\glui\build_x64\Debug
..\..\Dependencies\freeglut-3.4.0\build_x64\lib\Debug
..\..\Dependencies\glew-2.1.0\lib\Release\x64
..\..\Dependencies\opencv\build\x64\vc16\lib
```

You will find the relative paths to the lib folders of the dependencies we are using. These will be:

..\..\Dependencies\glui\build_x64\Debug

..\..\Dependencies\freeglut-3.4.0\build_x64\lib\Debug

..\..\Dependencies\glew-2.1.0\lib\Release\x64

..\..\Dependencies\opencv\build\x64\vc16\lib

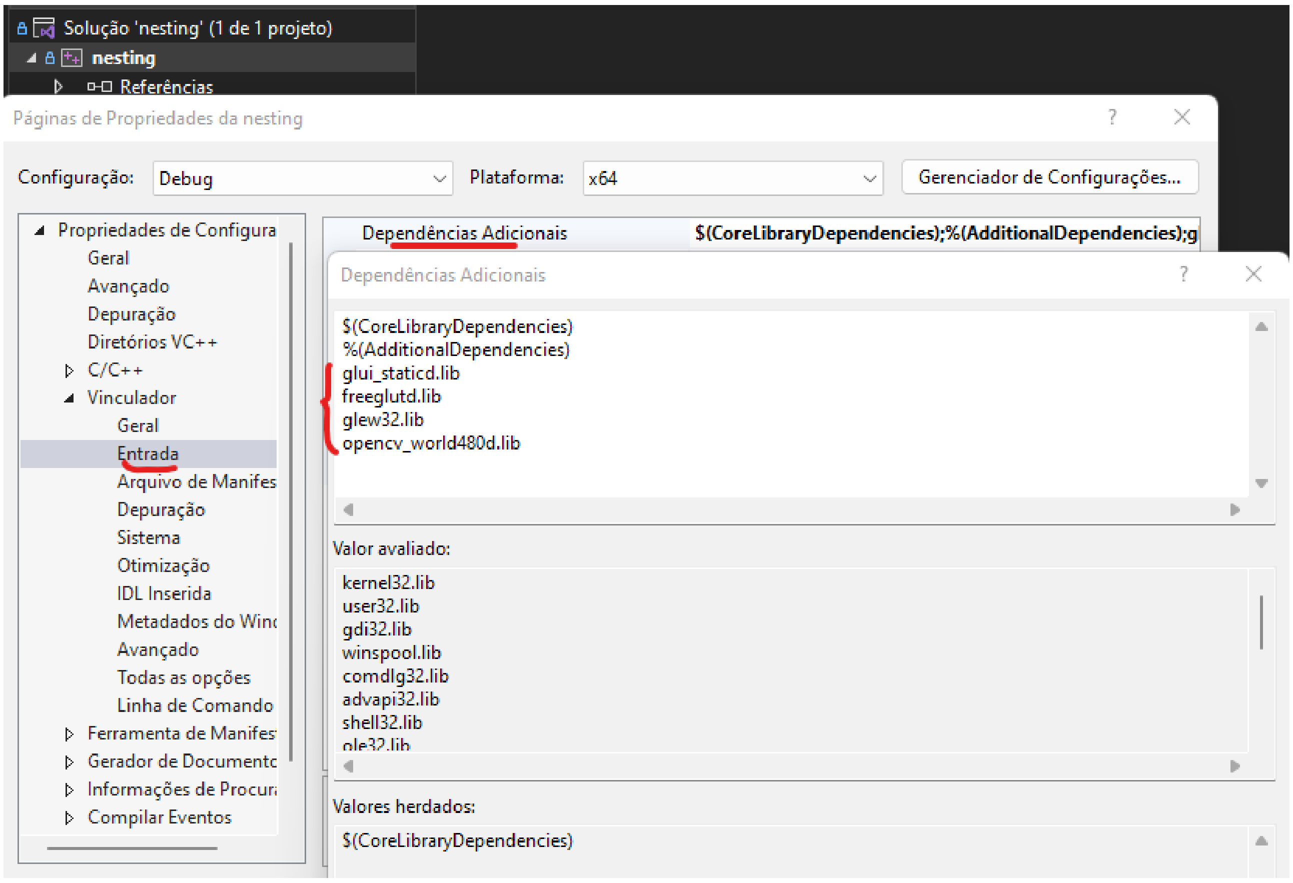
(Note that glui and freeglut paths point to Debug folders. Had you been editing the Release configuration, they should point to the Release ones.)

Confirm that these paths effectively correspond to the include folders provided by the libraries you downloaded. Note that, **if you downloaded different versions of the dependencies, you must change the paths to match your versions.**

3. Verify that you have specified the correct libraries names

Go back once again to the properties window and, in the "Linker" category, choose the option "Input". On the right frame you will find an entry called "Additional Dependencies" click on the arrow to edit this.

Your screen should look like this:



Dependências Adicionais: $(CoreLibraryDependencies);%(AdditionalDependencies);g

```
$(CoreLibraryDependencies)
%(AdditionalDependencies)
glui_staticd.lib
freeglutd.lib
glew32.lib
opencv_world480d.lib
```

Valor avaliado:

```
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
advapi32.lib
shell32.lib
ole32.lib
```

Valores herdados:

```
$(CoreLibraryDependencies)
```

You will find the names of the lib folders of the dependencies we are using. These will be:

glui_staticd.lib

freeglutd.lib

glew32.lib

opencv_world480d.lib

Note that all of this libraries (except glew32.lib) end with a ‘d’. This is because they are debug libraries. Had you been editing the Release configuration, you should drop the ‘d’ (ex. glui_static.lib).

Now you should be able to compile the project successfully.

To run it you still have to execute the step below.

4. Copy the dll’s into project folder

To be able to run the project, the executable must be able to find the dll’s it needs. For this reason I’ve placed the required dll’s into the project folder: **repos_demo\nesting2\nesting.**

You should copy the following dll’s into project folder:

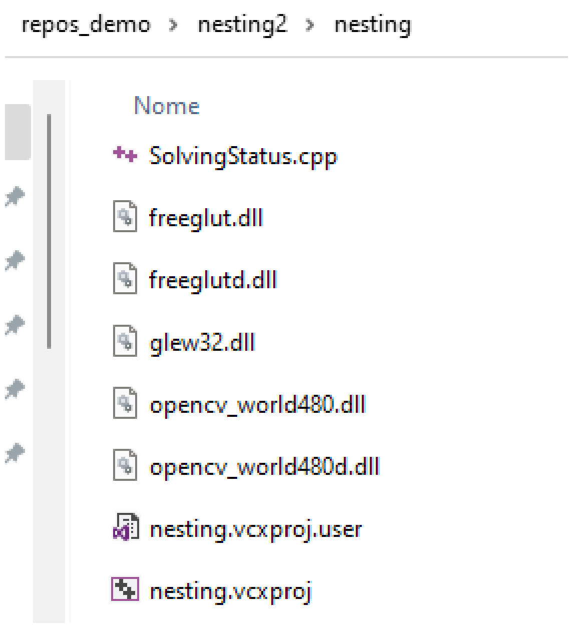
freeglutd.dll -> from **repos_demo\Dependencies\freeglut-3.4.0\build_x64\bin\Debug**

freeglut.dll -> from **repos_demo\Dependencies\freeglut-3.4.0\build_x64\bin\Releas**

glew32.dll -> from **repos_demo\Dependencies\glew-2.1.0\bin\Release\x64**

opencv_world480.dll and opencv_world480d.dll -> from **repos_demo\Dependencies\opencv\build\x64\vc16\bin**

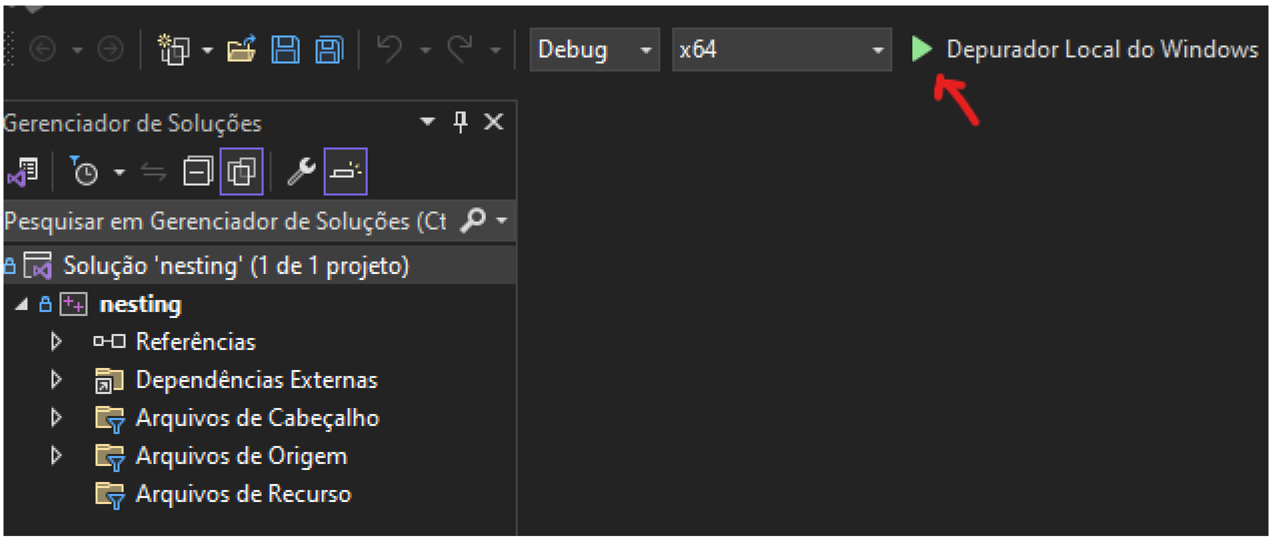
Your project folder should look like this:


repos_demo › nesting2 › nesting

Nome
SolvingStatus.cpp
freeglut.dll
freeglutd.dll
glew32.dll
opencv_world480.dll
opencv_world480d.dll
nesting.vcxproj.user
nesting.vcxproj

5. Compile and run the project

Now you should be able to build and run the project, as below:

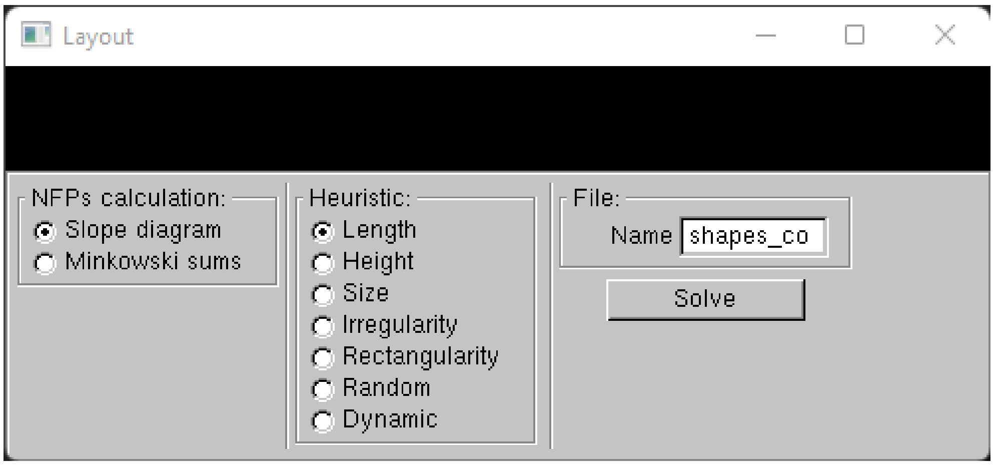

Debug
x64
Depurador Local do Windows

Gerenciador de Soluções

Pesquisar em Gerenciador de Soluções (Ct

Solução 'nesting' (1 de 1 projeto)
- nesting
  - Referências
  - Dependências Externas
  - Arquivos de Cabeçalho
  - Arquivos de Origem
  - Arquivos de Recurso

This is what a successful run looks like:

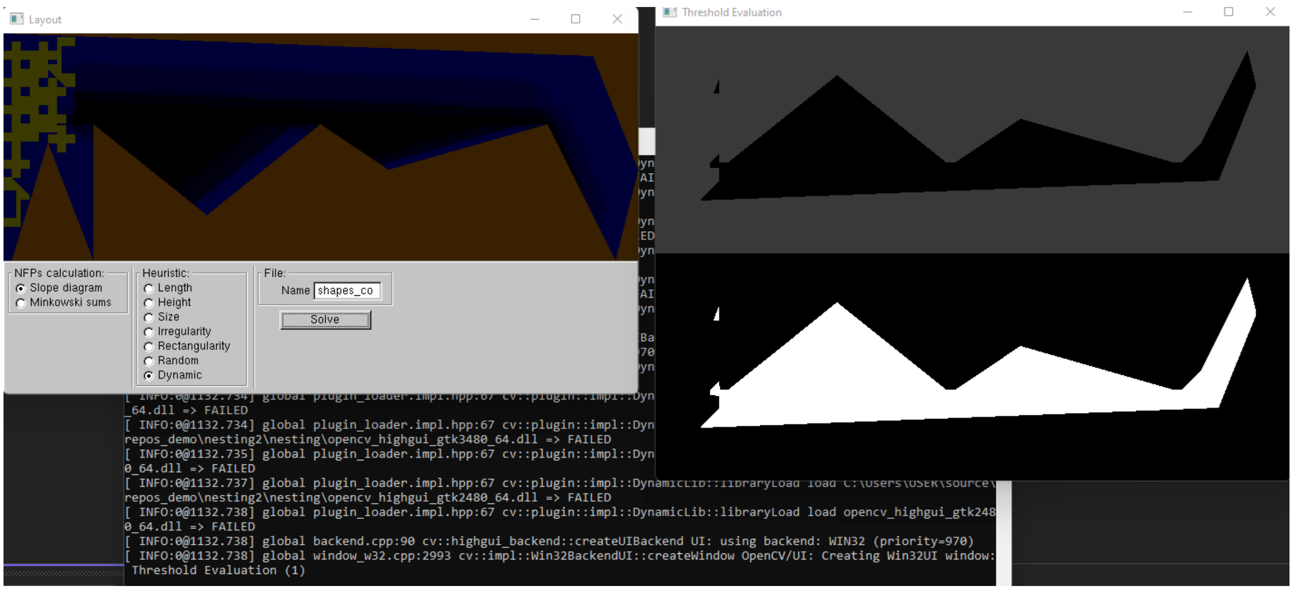

Layout

NFPs calculation:
[✓ Slope diagram]
Minkowski sums

Heuristic:
[✓ Length]
Height
Size
Irregularity
Rectangularity
Random
Dynamic

File:
Name: shapes_co

If you try it out, it should look like this:



NFPs calculation:
[✓ Slope diagram]
Minkowski sums

Heuristic:
Length
Height
Size
Irregularity
Rectangularity
Random
[✓ Dynamic]

File:
Name: shapes_co

```
[ INFO:0@1132.734] global plugin_loader.impl.hpp:67 cv::plugin::impl::Dyn
_64.dll => FAILED
[ INFO:0@1132.734] global plugin_loader.impl.hpp:67 cv::plugin::impl::Dyn
repos_demo\nesting2\nesting\opencv_highgui_gtk3480_64.dll => FAILED
[ INFO:0@1132.735] global plugin_loader.impl.hpp:67 cv::plugin::impl::Dyn
0_64.dll => FAILED
[ INFO:0@1132.737] global plugin_loader.impl.hpp:67 cv::plugin::impl::DynamicLib::libraryLoad load C:\Users\USER\source\
repos_demo\nesting2\nesting\opencv_highgui_gtk2480_64.dll => FAILED
[ INFO:0@1132.738] global plugin_loader.impl.hpp:67 cv::plugin::impl::DynamicLib::libraryLoad load opencv_highgui_gtk248
0_64.dll => FAILED
[ INFO:0@1132.738] global backend.cpp:90 cv::highgui_backend::createUIBackend UI: using backend: WIN32 (priority=970)
[ INFO:0@1132.738] global window_w32.cpp:2993 cv::impl::Win32BackendUI::createWindow OpenCV/UI: Creating Win32UI window:
 Threshold Evaluation (1)
```