1. Setup Clion and create a new project in the same way you do in PyCharm.
2. [OPTIONAL] In the New Project window, select C++17 or C++20 as the language standard to be able to take advantage of the newer and more-user friendly features of the language.
3. Once the project has been launched, click on the drop-down menu with the word "Debug" in the top taskbar and select Edit CMake Profiles...
4. On the settings menu that appears in a new window, click on the plus button in the Profiles section (Fig. 4.1). The Release profile should be added to the project automatically (Fig. 4.2). Once done, click on the blue OK button on the bottom left corner of the window. A highlighted folder named "cmake-build-release" should be visible in the project overview (Fig. 4.3).
5. Click on the three-dot button in the vertical sidebar on the left side of the screen and select Vcpkg (Fig. 5.1).
6. On the menu that appears at the bottom of the screen (Fig. 5.1), click on the plus button, select the "Add vcpkg integration to existing CMake profiles" in case it is not already enabled, and make sure that both Debug and Release profile have been selected (Fig. 5.2). Once done, click on the blue OK button on the bottom left corner of the window.
7. Once the spawned processes have finished (i.e., when the blue progress bar on the bottom right of the screen disappears), you should see a "Clone and bootstrap successful message" on the bottom left side of the screen, next to the vcpkg icon, and the icon next to the name of your vcpkg installation will have turned from gray to orange (Fig. 7.1).
8. Click on the name of your vcpkg installation and search for "cgal" on the search bar that appears to its right. Select the package name and click on "Install" (Fig. 8.1). Make sure that the triplet option has been set to "Let vcpkg decide".

   [WARNING] This process will take some time. Make sure that you have a stable internet connection throughout and that your PC does not go to sleep or turn off.

9. Once the spawned processes have finished (i.e., when the blue progress bar on the bottom right of the screen disappears), you should see an "Install successful. Installed packages: CGAL" on the bottom left side of the screen, next to the vcpkg, along with a prompt to add CGAL to CMakeLists.txt, which you can think of as a configuration or manifest file for your project. For instance, this is which version of C++ you're using and which are your library dependencies.

   [WARNING] Do not use the prompt!

10. Once CGAL has been installed, the menu you used to install it should mention which triplet is present in your system, along with the library version (Fig. 10.1).
11. Select Edit CMake Profiles... (Step 3), and *for each* profile, click on the Cache variables drop-down menu and replace CVPKG_TARGET_TRIPLET with the triplet from Step 10 (Fig. 11.1). Once done, click on the blue OK button on the bottom left corner of the window.
12. Open CMakeLists.txt and append the following commands to the file:

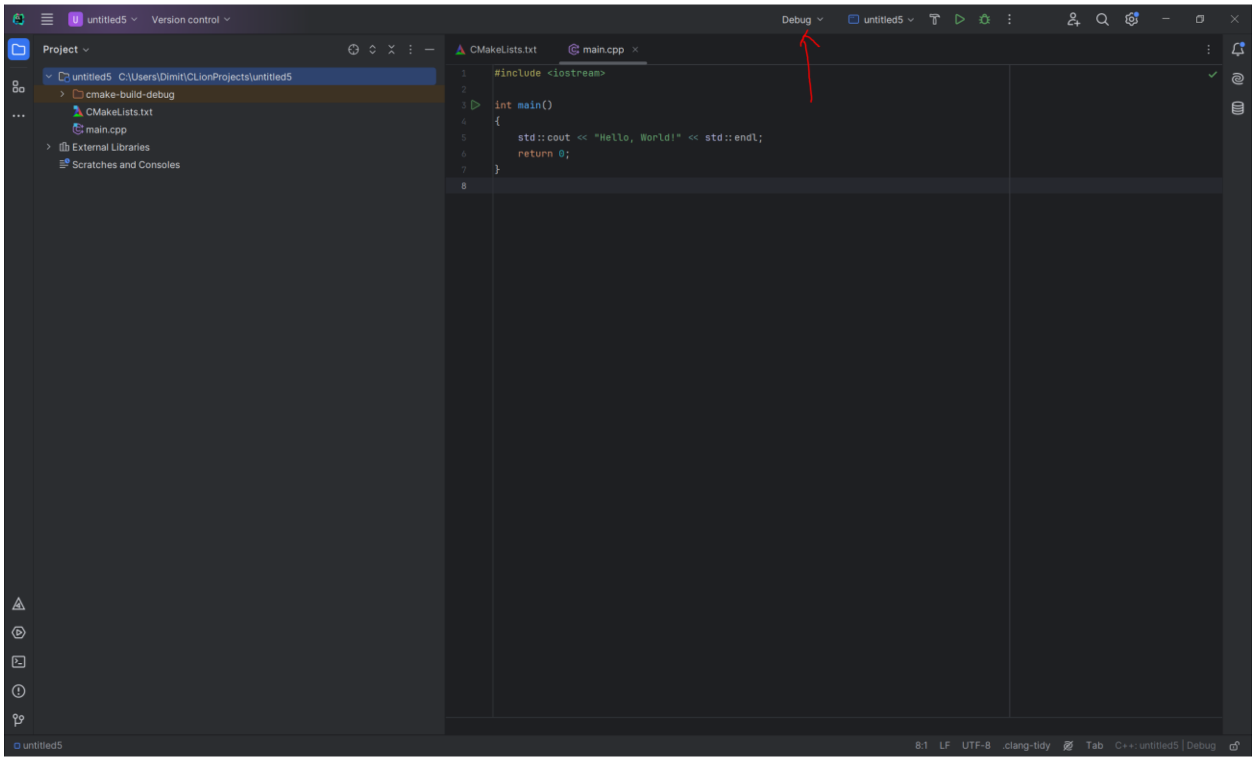
    ```
    find_package(CGAL CONFIG REQUIRED)
    target_link_libraries(<project_name> PRIVATE CGAL::CGAL)
    ```

where <project_name> is the name of your project as you defined it and appears next to the profiles drop-down menu, the project overview, as well as the second line of CMakeLists.txt (Fig 12.1).

13. Once done, press the circled button on the top right of Fig. 12.1 to reload your CMake changes. You can also do this be pressing Ctrl+Shift+O. The project should now rebuild for both profiles. The release build should compile without errors, whereas in the debug build, CGAL will complain about potential performance issues. This is expected behavior (Fig. 13.1a-b).

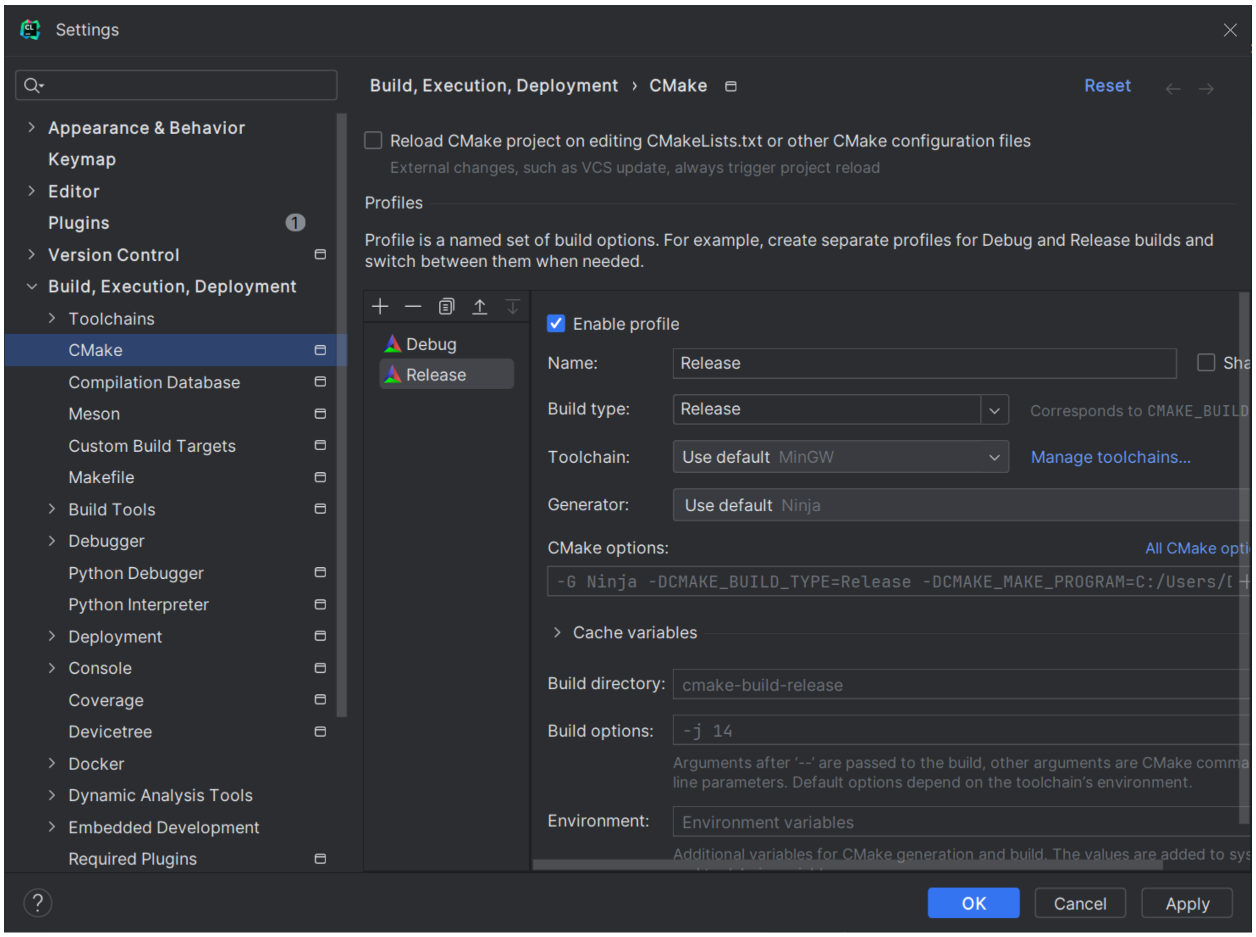
    [NOTE] You may get a CMake warning about an unused, manually specified variable. This is fine to ignore (Fig. 13.1b).

14. Copy and paste this example into main.cpp and make sure it compiles and runs in both release and debug.



```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Fig. 4.1



Build, Execution, Deployment › CMake

☐ Reload CMake project on editing CMakeLists.txt or other CMake configuration files

External changes, such as VCS update, always trigger project reload

Profiles

Profile is a named set of build options. For example, create separate profiles for Debug and Release builds and switch between them when needed.

✓ Enable profile

Name:	Release	☐ Sha
Build type:	Release	Corresponds to CMAKE_BUILD
Toolchain:	Use default MinGW	
Generator:	Use default Ninja	

CMake options:

```
-G Ninja -DCMAKE_BUILD_TYPE=Release -DCMAKE_MAKE_PROGRAM=C:/Users/[
```

› Cache variables

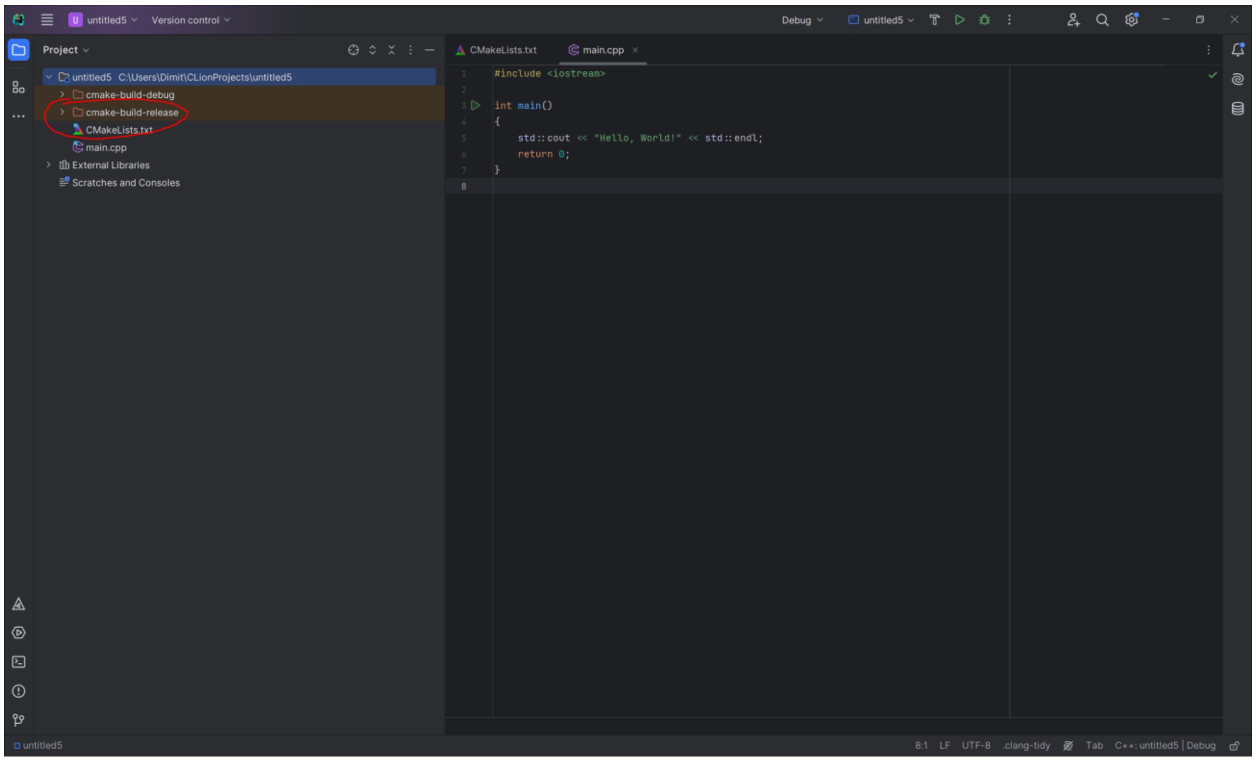
Build directory:	cmake-build-release
Build options:	-j 14

Arguments after '--' are passed to the build, other arguments are CMake comma line parameters. Default options depend on the toolchain's environment.

Environment: Environment variables

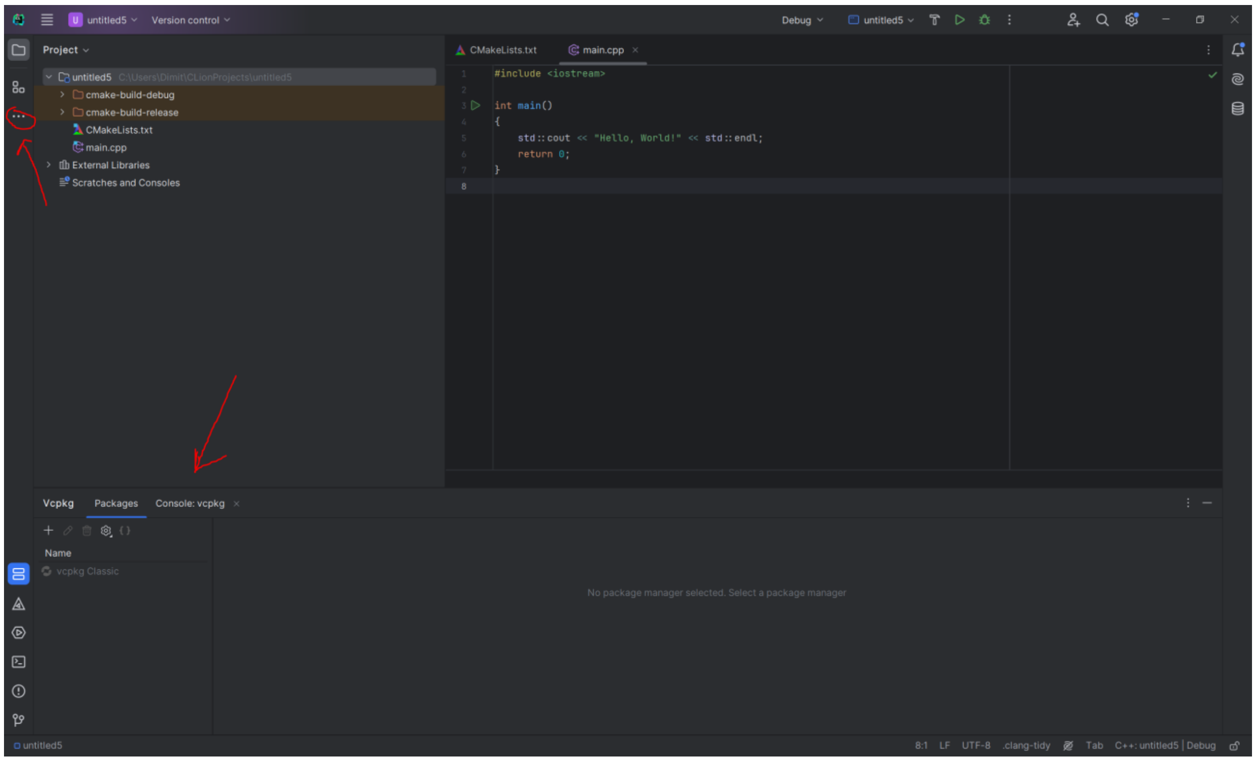
Additional variables for CMake generation and build. The values are added to sys

Fig. 4.2



```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

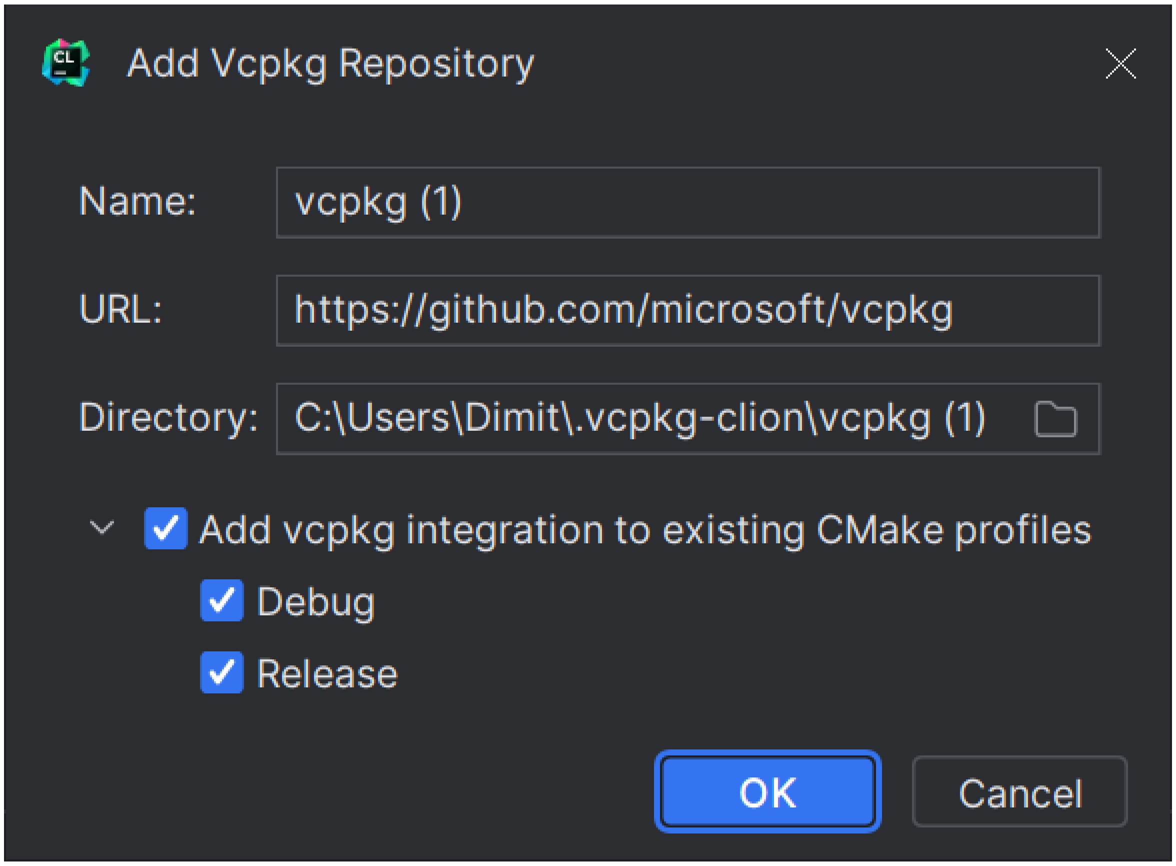
Fig. 4.3



```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

No package manager selected. Select a package manager

Fig. 5.1


Add Vcpkg Repository

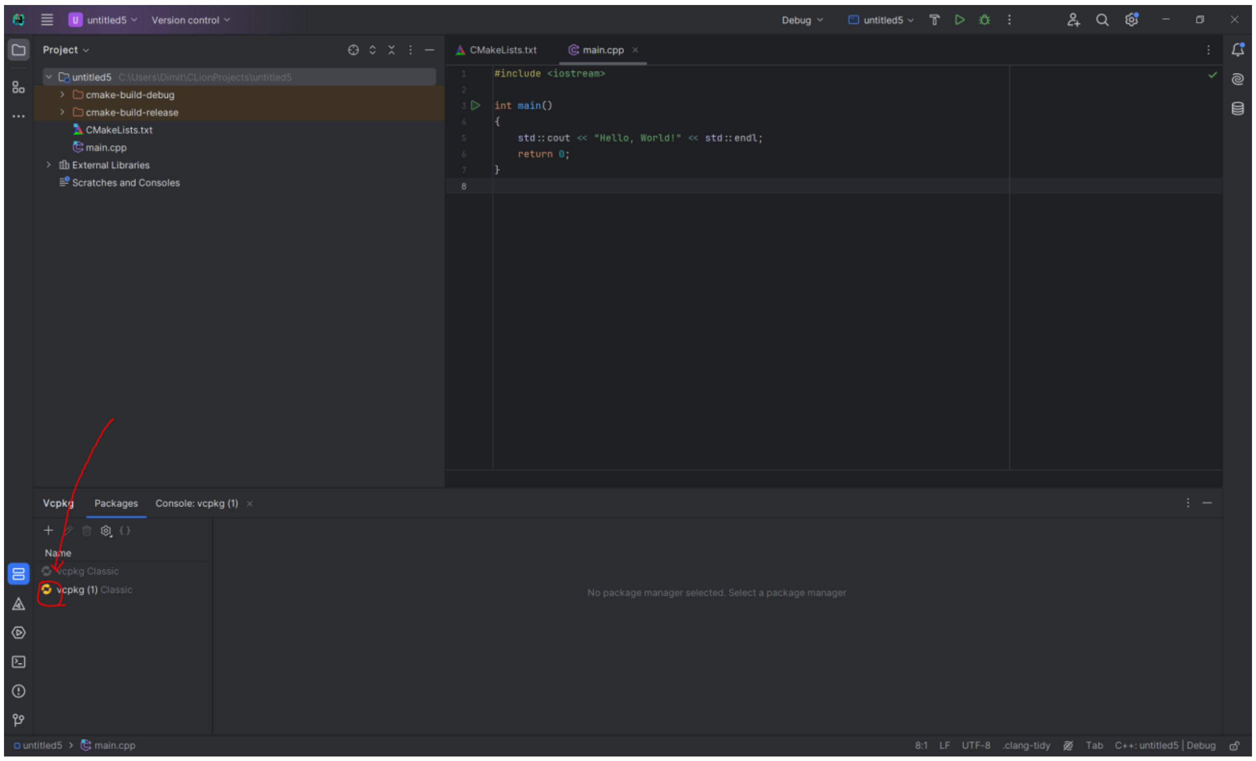
Name: vcpkg (1)

URL: https://github.com/microsoft/vcpkg

Directory: C:\Users\Dimit\.vcpkg-clion\vcpkg (1)

- [✓] Add vcpkg integration to existing CMake profiles
  - [✓] Debug
  - [✓] Release

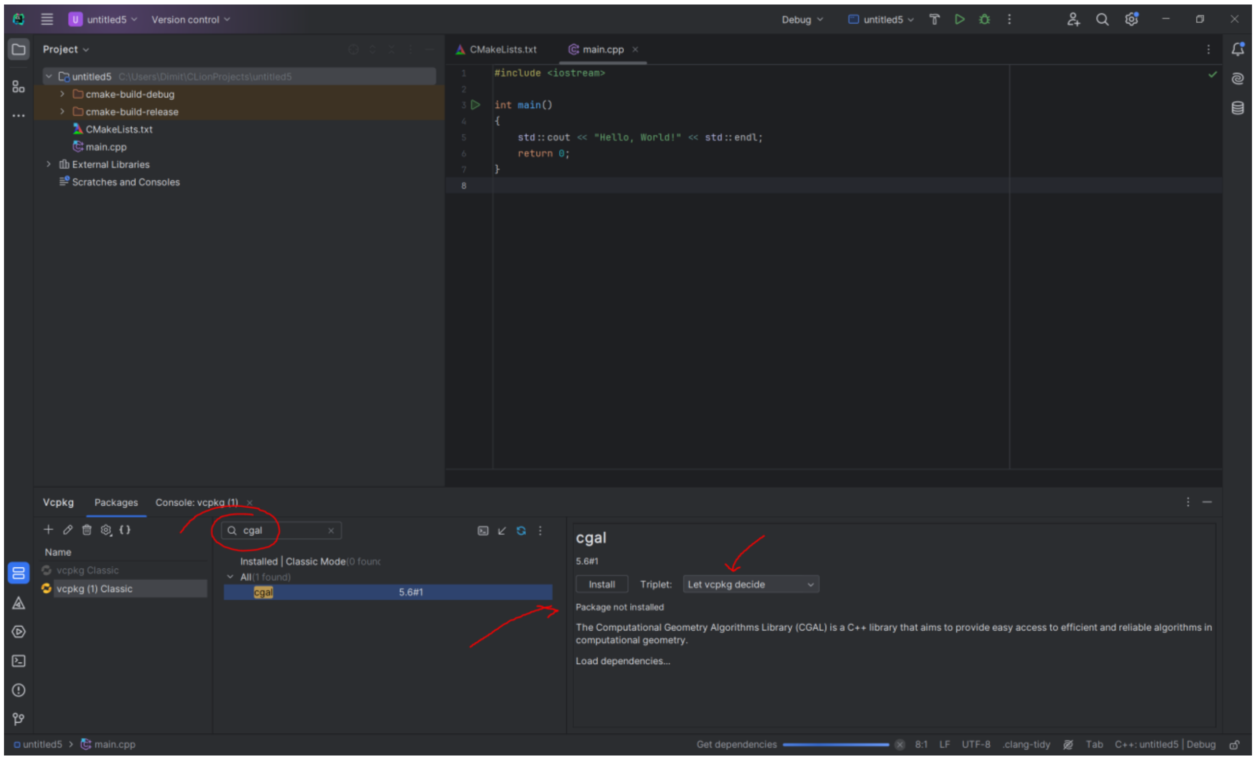
Fig. 5.2



```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

No package manager selected. Select a package manager

Fig. 7.1



```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

cgal

5.6#1

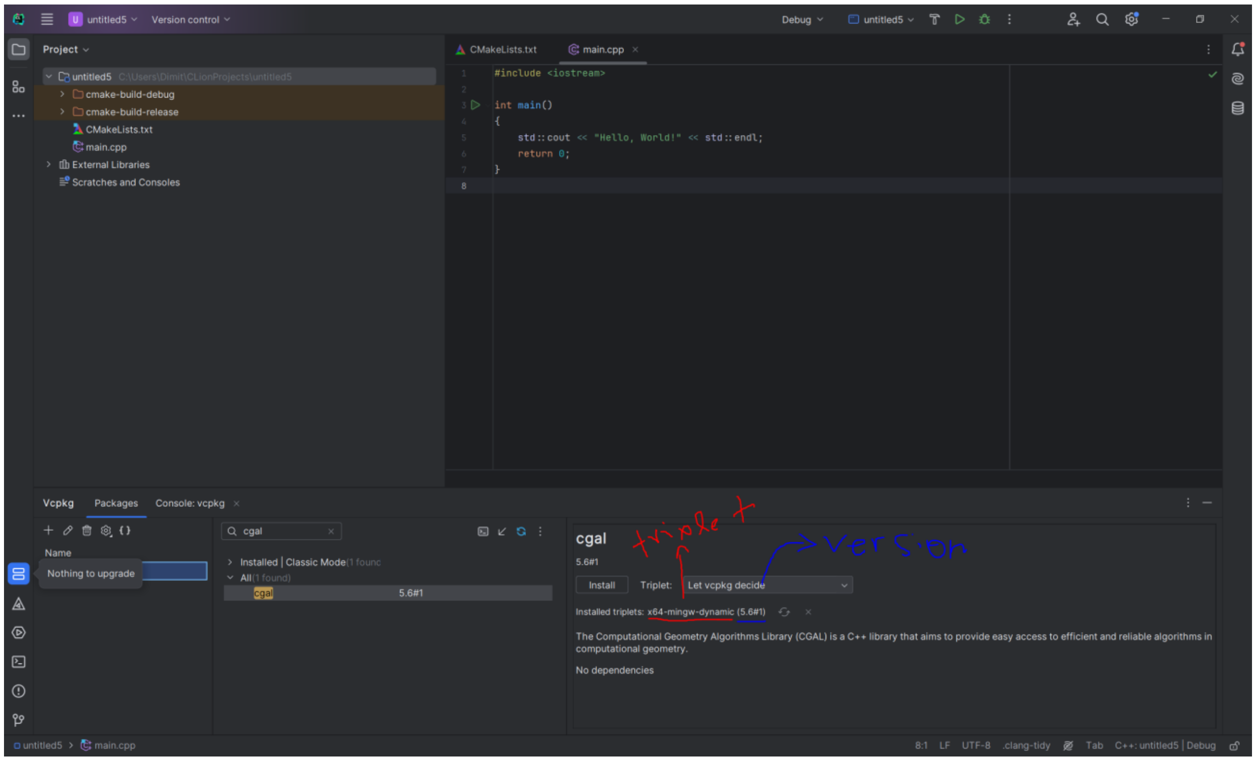
Triplet: Let vcpkg decide

Package not installed

The Computational Geometry Algorithms Library (CGAL) is a C++ library that aims to provide easy access to efficient and reliable algorithms in computational geometry.

Load dependencies...

Fig. 8.1



untitled5 C:\Users\Dimit\CLionProjects\untitled5

```
#include <iostream>

int main()
{
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

Nothing to upgrade

cgal

Installed | Classic Mode(1 found)
All(1 found)
cgal 5.6#1

cgal
5.6#1

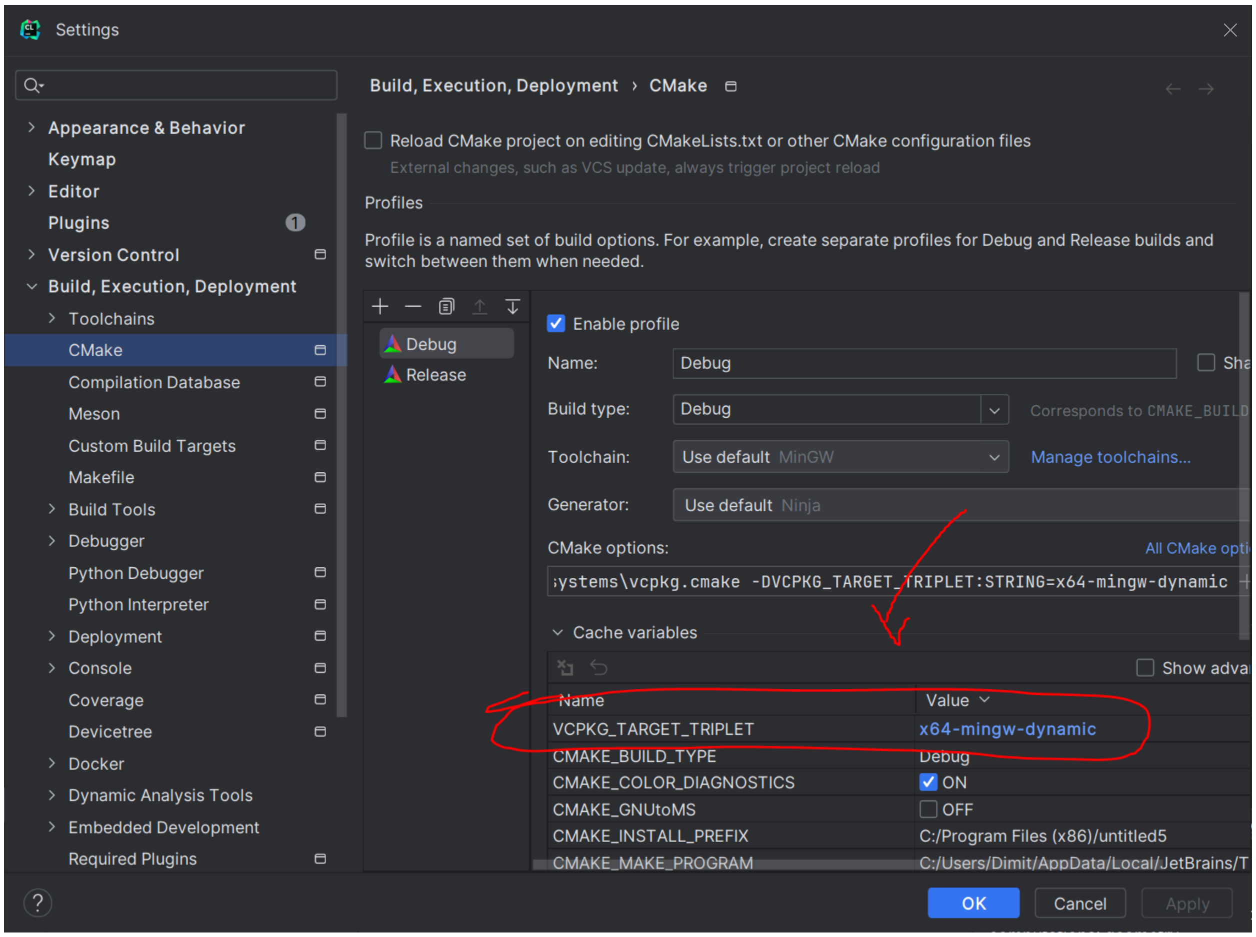
Triplet: Let vcpkg decide

Installed triplets: x64-mingw-dynamic (5.6#1)

The Computational Geometry Algorithms Library (CGAL) is a C++ library that aims to provide easy access to efficient and reliable algorithms in computational geometry.

No dependencies

Fig. 10.1



Build, Execution, Deployment › CMake

Reload CMake project on editing CMakeLists.txt or other CMake configuration files
External changes, such as VCS update, always trigger project reload

Profiles

Profile is a named set of build options. For example, create separate profiles for Debug and Release builds and switch between them when needed.

Debug
Release

✓ Enable profile

Name: Debug

Build type: Debug — Corresponds to CMAKE_BUILD

Toolchain: Use default MinGW — Manage toolchains...

Generator: Use default Ninja

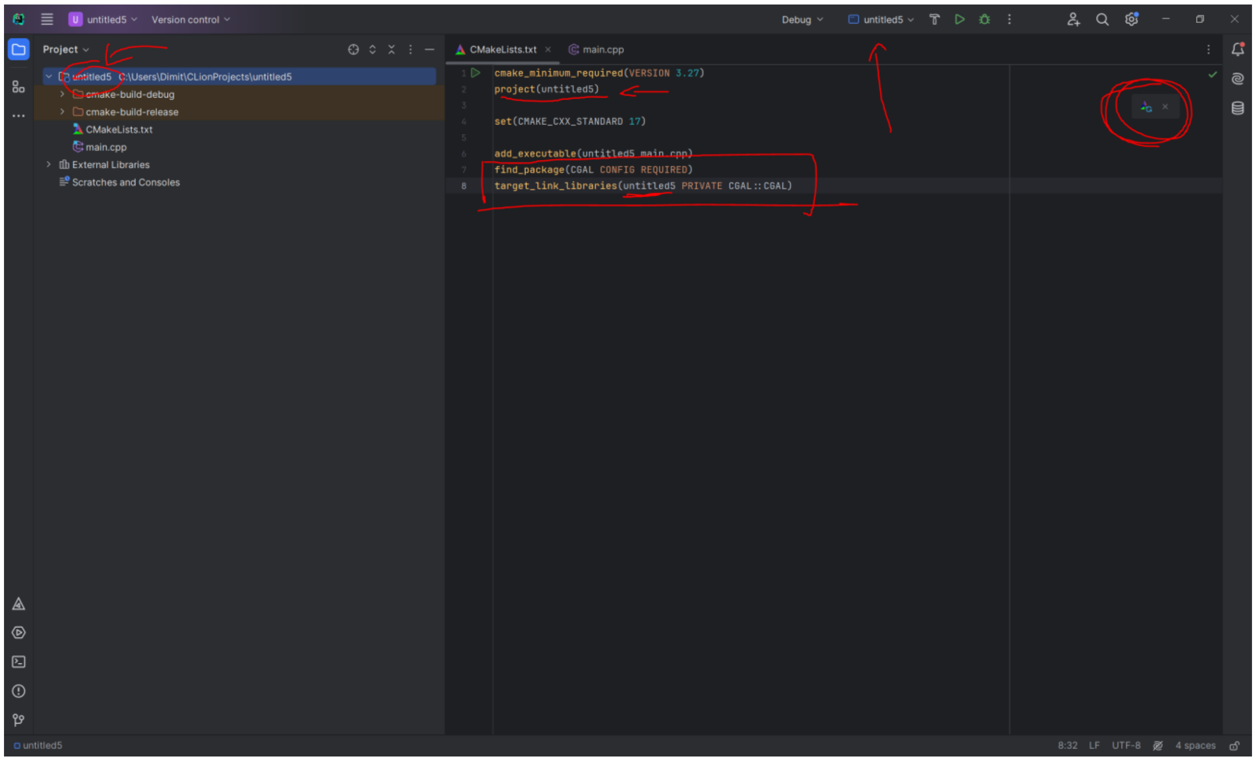
CMake options: All CMake opti

```
;ystems\vcpkg.cmake -DVCPKG_TARGET_TRIPLET:STRING=x64-mingw-dynamic
```

Cache variables

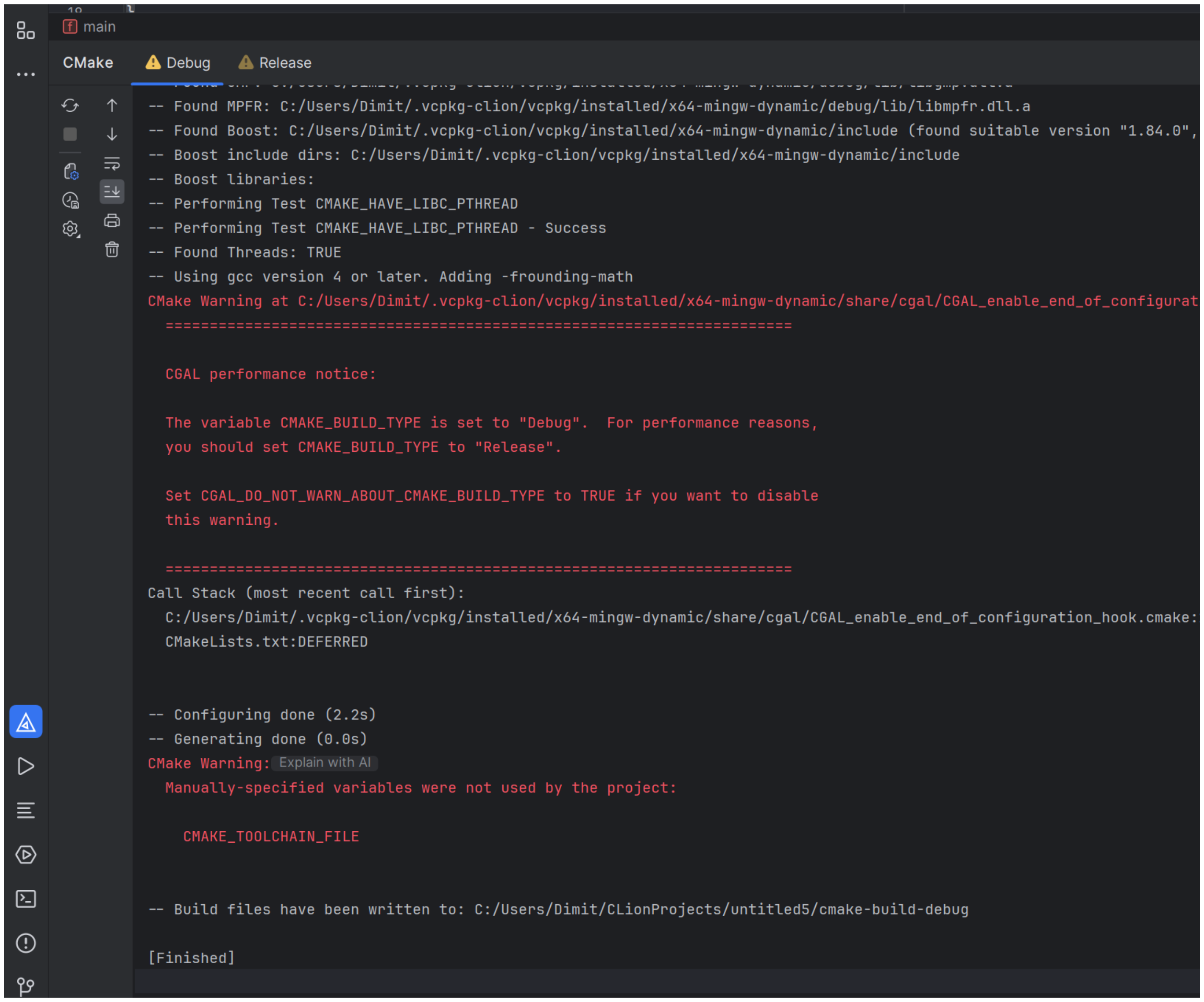
Name	Value
VCPKG_TARGET_TRIPLET	x64-mingw-dynamic
CMAKE_BUILD_TYPE	Debug
CMAKE_COLOR_DIAGNOSTICS	✓ ON
CMAKE_GNUtoMS	OFF
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/untitled5
CMAKE_MAKE_PROGRAM	C:/Users/Dimit/AppData/Local/JetBrains/T

Fig 11.1



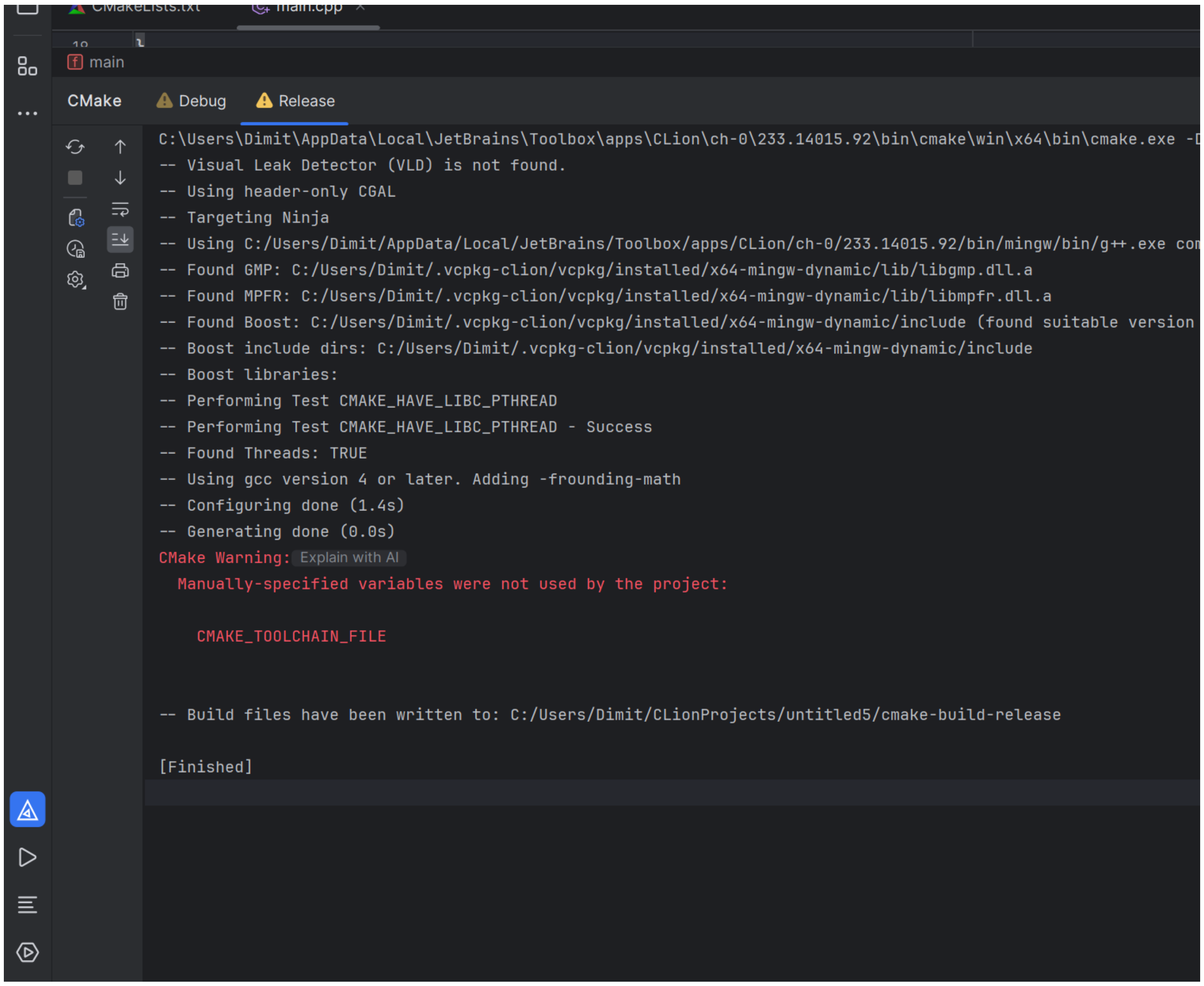
```
cmake_minimum_required(VERSION 3.27)
project(untitled5)

set(CMAKE_CXX_STANDARD 17)

add_executable(untitled5 main.cpp)
find_package(CGAL CONFIG REQUIRED)
target_link_libraries(untitled5 PRIVATE CGAL::CGAL)
```

Fig 12.1



```
-- Found MPFR: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/debug/lib/libmpfr.dll.a
-- Found Boost: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/include (found suitable version "1.84.0",
-- Boost include dirs: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/include
-- Boost libraries:
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Using gcc version 4 or later. Adding -frounding-math
CMake Warning at C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/share/cgal/CGAL_enable_end_of_configurat
  =======================================================================

  CGAL performance notice:

  The variable CMAKE_BUILD_TYPE is set to "Debug".  For performance reasons,
  you should set CMAKE_BUILD_TYPE to "Release".

  Set CGAL_DO_NOT_WARN_ABOUT_CMAKE_BUILD_TYPE to TRUE if you want to disable
  this warning.

  =======================================================================
Call Stack (most recent call first):
  C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/share/cgal/CGAL_enable_end_of_configuration_hook.cmake:
  CMakeLists.txt:DEFERRED


-- Configuring done (2.2s)
-- Generating done (0.0s)
CMake Warning: Explain with AI
  Manually-specified variables were not used by the project:

    CMAKE_TOOLCHAIN_FILE


-- Build files have been written to: C:/Users/Dimit/CLionProjects/untitled5/cmake-build-debug

[Finished]
```

Fig 13.1a



```
C:\Users\Dimit\AppData\Local\JetBrains\Toolbox\apps\CLion\ch-0\233.14015.92\bin\cmake\win\x64\bin\cmake.exe -D
-- Visual Leak Detector (VLD) is not found.
-- Using header-only CGAL
-- Targeting Ninja
-- Using C:/Users/Dimit/AppData/Local/JetBrains/Toolbox/apps/CLion/ch-0/233.14015.92/bin/mingw/bin/g++.exe com
-- Found GMP: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/lib/libgmp.dll.a
-- Found MPFR: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/lib/libmpfr.dll.a
-- Found Boost: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/include (found suitable version
-- Boost include dirs: C:/Users/Dimit/.vcpkg-clion/vcpkg/installed/x64-mingw-dynamic/include
-- Boost libraries:
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Using gcc version 4 or later. Adding -frounding-math
-- Configuring done (1.4s)
-- Generating done (0.0s)
CMake Warning:
  Manually-specified variables were not used by the project:

    CMAKE_TOOLCHAIN_FILE


-- Build files have been written to: C:/Users/Dimit/CLionProjects/untitled5/cmake-build-release

[Finished]
```

Fig 13.1b